Automation of vehicular systems using deep reinforcement learning and mean-field models: Application to heavy duty trucks

by

Saleh Mahdi Albeaik

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexandre M. Bayen, Chair
Doctor Xiao-Yun Lu
Professor Alexander Skabardonis
Professor Laurent El Ghaoui

Summer 2020

Automation of vehicular systems using deep reinforcement learning and mean-field models:
Application to heavy duty trucks

Abstract

Automation of vehicular systems using deep reinforcement learning and mean-field models: Application to heavy duty trucks

by

Saleh Mahdi Albeaik

Doctor of Philosophy in Engineering - Civil and Environmental Engineering

University of California, Berkeley

Professor Alexandre M. Bayen, Chair

The transportation sector consumes about a third of all energy consumed in the world, about a third of which is consumed by trucks. Future transportation systems must address this energy challenge, in addition to the other inefficiencies related to time, money, and lives lost while the system is operating. Vehicle automation is one of the promising opportunities underway. For instance, cooperative adaptive cruise control, an extension of the more popular cruise control and adaptive cruise control systems, promises to reduce fuel consumption by up to 15% for participating trucks, reduce emissions, increase road capacity at high technology penetration rates, and contribute to road safety.

Heavy duty trucks are complex vehicles that are designed and built for specific mission requirements. Any of these trucks could be equipped from a wide selection of vehicle components with a significantly wide spectrum of operating dynamics and performances. Driving a heavy duty truck is an equally complex task. Human drivers must be well educated and trained about the specific truck they are about to drive and operate. They must optimize in real-time for factors such as truck dynamics and driving performance; road, truck, and payload safety; truck operation economics; truck driving law constraints; mission constraints; in addition to background traffic on the road.

Automation of heavy duty truck operation tasks require equally advanced engineering tools. For instance, high precision modeling and control have historically required a detailed study of each subject truck. This thesis presents a process using deep learning and deep reinforcement learning for microscopic longitudinal modeling and control of such trucks that is agnostic to their internal mechanics. The process is demonstrated and evaluated for several truck mechanical configurations using high fidelity simulation and in the field. Cruise control of single truck operations has been considered, in addition to cooperative adaptive cruise control for multi-truck coordination.

Long haul heavy duty trucks often drive within shared road infrastructure with background traffic. To account for this traffic on the road, we consider multi-scale partial differential equation mean-field models. With this approach, each truck is modeled microscopically while background traffic is modeled mesoscopically. A nondissipative numerical solver is developed and evaluated for computational study of these models. The solver maintains structure and resolution at a wide range of discretization resolutions suitable for development of optimal control laws.

This thesis investigates computational methods for the automation of heavy duty trucks. While vehicle driving automation is already underway, more investigation is still required to bring about full autonomy. The future of the transportation system and trucking could benefit from further study and development of the sciences and engineering of autonomy with consideration to the complex interplay between the vehicle as an agent, the transportation system as an operations context, the logistics system as a mission context, and the human beneficiary.

لمن علمني حرفا او مزيد..

To those who have had contributed to my learning, education, and growth.

Contents

List of Figures

List of Tables

Acknowledgments

A tale of two institutions

A three year PhD is too brief except for the richness and density of learning, education, and growth experiences it had offered. Only a tiny fraction of these three eventful years is suitable for thesis publication, and only a smaller fraction I yet have a sufficient language, recall, and number of pages to articulate and document before you in this acknowledgement section.

Carl Jung, the founder of analytical psychology, argued for the archetypes as shared universal symbols of the human experience. Joseph Campbell, a comparative mythologist, maintained that mythical stories articulate the gist of the human experience, and argued for the archetypal hero's journey, an outline of a monomyth that appears universally applicable across all successful world mythical journeys. Campbell's monomyth is the closest to a satisfactory summary of this three year journey I had come to learn about by the time of publishing of this manuscript. Nothing but a curious wanderer myself, I would like to acknowledge and express my deepest gratitude to the true heroes of my journey, those whose names appear here, and most especially, those who were always there for me in the background.

> The hero, therefore, is the man or woman who has been able to battle past his personal and local historical limitations to the generally valid, normally human forms. Such a one's visions, ideas, and inspirations come pristine from the primary springs of human life and thought.
>
> A hero ventures forth from the world of common day into a region of supernatural wonder: fabulous forces are there encountered and a decisive victory is won: the hero comes back from this mysterious adventure with the power to bestow boons on his fellow man.
>
> Whether the hero be ridiculous or sublime, Greek or barbarian, gentile or Jew, his journey varies little in essential plan. Popular tales represent the heroic action as physical; the higher religions show the deed to be moral; nevertheless, there will be found astonishingly little variation in the morphology of the adventure, the character roles involved, the victories gained.
>
> *Joseph Campbell, 1949*

I am greatly indebted to Alex Bayen and Anas Alfares, the two prominent guides and magical helpers prior to and throughout this journey. I am thankful to Xiao-Yun Lu, Hatim Bukhari, Mansour Alsaleh, and Sattam Alsubaiee for picking up the baton, day to day, from day zero of the joint effort between UC Berkeley and King Abdulaziz City for Science and Technology (KACST) to plant the first seed for autonomous truck research and development in Saudi Arabia, the primary component of this PhD.

I am also thankful to the first to believe in the potential of this effort and commit to contribute, the applied vehicle dynamics and automation team at California PATH, John Spring, David Nelson, Fang-Chieh Chou, and Abdul Rahman Kreidieh; the team at Zahid Tractor, Ahmed Ismail, Abdullah Ossabi, and Mohammed Mustafa; Tariq Aldiweesh, Ammar Hazzazi, Helen Bassham, and the rest of the management and administrative staff from each of these institutions; and the many others who made success possible.

The first fruits of this effort would have been impossible to reap without the tremendous help from the Communication and Information Technology Research Institute team at KACST, Khalid Alshareef, Fadwa Alaskar, Mohammed Aloqayli, Basem Hassan, Radwan Noor, Majed Alghamdi, and Mohammad Khorsheed; and students, postdocs, visiting scholars and engineers including Zhanwen Liu, Carlos Flores, and Sulaiman Almatrudi.

I am grateful for the opportunity to work with and learn from the theoretics and mathematics team from UC Berkeley and Rutgers University, Alex Keimer, Amaury Hayat, Xiaoqian Gong, Sean McQuade, Yiling You, Nicolas Kardous, and Benedetto Piccoli.

The ground for the major PhD milestones, the navigational aid, and all the feedback and support from faculty, Alex Bayen, Xiao-Yun Lu, Alex Skabardonis, Laurent El Ghaoui, Steven Glaser, Michael Cassidy, Jasenka Rakas, Marta González, and Mark Hansen; advising and administrative staff, Shelley Okimoto and Helen Bassham; then transportation engineering department seniors, David Kan, Paul Anderson, and Mostafa Harb; friends and colleagues, Hussain AlSalem, Alex Keimer, Fangyu Wu, Fang-Chieh Chou, Yiling You, Marsalis Gibson, Ashkan Yousefpour, Eugene Vinitsky, Jessica Lazarus, Soomin Woo, Juan Caicedo, Bilal El-Halabi, and Ibrahim Itani; and the many students, postdocs and visiting scholars who were central to the whole experience at Mobile Sensing Lab; from Civil and Environmental Engineering, Mechanical Engineering, Electrical Engineering and Computer Science, and Mathematics departments was indispensable. The undergraduate research assistant team gave me all the leverage I needed to juggle all this workload; Trevor Wu, Ganeshnikhil Vurimi, Sarah Bhaskaran, Henry Leou, and Armittage Annang.

This journey would have been meaningless without all the people in the background; all my family and friends available for me around the clock at Qatif, Jeddah, Riyadh, Tokyo, Penang, Vienna, London, Boston, New York, the Happy Valley, San Diego, Irvine, Los Angeles, and off course the Bay Area. At the Bay Area and around Berkeley, I am lucky to have met, and made some of the most precious memories with, friends of whom were my portal to the world, and others who were my connection to home.

My journey was sharpened and refined by the help and support of many other guides and magical helpers in the background including Andrej, Scott, and Rich. My Bay Area student experience would have never been complete without the morning coffees, the afternoon lunches, and the evening ice-creams; to those who made them for me day after day, thank you! I cannot conclude this section without expressing my special appreciation to Michael Marchant and his family for generously, respectfully, and patiently hosting my residency at Berkeley.

This collective journey was generously funded by KACST, and was hosted and supported at UC Berkeley by the Mobile Sensing Lab, Institute of Transportation Studies, California PATH, and department of Civil and Environmental Engineering, and at KACST by the Joint Centers of Excellence Program, Center for Complex Engineering Systems, Center Of Excellence for Telecom Applications, and Communication and Information Technology Research Institute.

This manuscript was written and concluded during the most unexpected and disturbing event of my generation; the COVID-19 pandemic. Dear human family, please heal, heal fast, and “do not go gentle into that good night” (Written by Dylan Thomas and echoed by Interstellar).

Chapter 1

Introduction

1.1 Introduction

The transportation system. The transportation system is one of the main large scale systems interfacing with virtually everyone's daily life and where significant scarce resource, such as time and energy, are being consumed. In the United States, 84 billion hours were spent on the road in 2015 accounting for an average of 45 minutes per capita per day [1]. This transportation system operates by consuming about 28% of total energy consumed in the country [2], and in 2016, $175 billion were spent by state and local governments on highways and roads [3].

Vehicle modeling and control. Some of these challenges are being promised to be alleviated by recent advances in automation, optimization, and emerging technologies such as connected automated vehicles, alternative fuels, and mobility operating models [4]–[8]. Studying transportation system efficiency and developing advanced technologies depend heavily on the availability of reliable vehicle models. This has raised interest in advanced tools for higher accuracy, detailed modeling, and easy of integration [9], [10].

Heavy duty trucks. The trucking industry alone consumes 10% of total national energy [11] and contribute 6% of all emissions to transport 70% of national economic value [12], [13]. Heavy duty trucks used for ground freight are complex vehicles and require advanced specialized engineering tools for modeling and automation [14]–[16]. Expected gains from automation can be significant. For instance, cooperative adaptive cruise control (CACC) is one of the promising technologies predicted to reduce fuel consumption by up to 15% of participating vehicles [17], and have the potential to double freeway capacity at high technology penetration rates [18].

This manuscript. This manuscript focuses on applications of computational methods for heavy duty truck (HDT) automation. Computational methods utilize computer simulations to solve complex engineering problems. Here, we use produce simulation models of heavy duty trucks using deep learning, develop truck control and coordination systems using simulation based deep reinforcement learning, and develop a numerical solver for multi-scale simulation of trucks and traffic flows.

1.2 Computational methods in science and engineering

Computational and data-driven methods. Computational methods have gained significant popularity that fueled numerous simulation-based science and engineering advances. Computational methods use numerical simulations instead of analytical and hand crafted solutions for science and engineering applications such as discovery and design [19]. Their use has the potential to simplify tedious scientific and engineering tasks. They can also help eliminate limiting assumptions/restrictions that are imposed for tractability or to encourage simplicity. Moreover, computational methods tend to support flexibility, scalability and problem-setup-complexity by design without the need for reformulation or specialization of solution. Sophisticated maneuvers, earth re-entry, and landing of space rockets are among the many recent breakthroughs for computational engineering methods [20]. The use of approximate surrogate models has been a standard practice in several computational fields such as optimization for aerospace applications [21]. The emergence of deep learning and its successes in these practices shows a promising path towards the next milestone; data-driven engineering from the ground up bypassing the physics equations used to develop the surrogate models.

Surrogate models reduce physics models into computationally tractable approximations. Surrogate models are light-weight data-based approximations of detailed physics models [21]. This model building approach is popular in literature such as computational optimization [21] and uncertainty quantification [22] for the reduction of computational cost of simulations. Computational cost of hand-engineered models of physics grow with the level of modeled details, and with numerical solver requirements such as conditioning and integration step-size. Models with cheaper simulation cost allow for bigger frameworks and more complex computational applications [21].

Deep Learning for surrogate modeling. Machine learning gained recent popularity for fitting highly accurate light-weight surrogate models. The authors of [23] demonstrated the use of deep neural networks to construct surrogate models for numerical simulators in the context of uncertainty propagation in stochastic elliptic partial differential equations. The authors of [24] described a method of constructing surrogates in the form of *support vector machine* (SVM) regressions for the purpose of exploring the parameter space of physiological models. The authors of [25] and [26] used machine learning techniques to construct surrogate models to simulate agent based models in the context of socioeconomic systems. In [27], the authors investigated the use of *Neural Networks* (NN) to build surrogate models for pavement construction payment-risk prediction models. The authors of [28] used neural-network-based approximation, sensitivity analysis, decomposition, and sampling techniques to develop a surrogate-based vehicle dynamic model. These machine and deep learning based surrogate models are expressive and computationally cheap for simulation intensive and real-time applications.

Deep and Surrogate models for discovery and design applications. Simulation-based computational methods have been effective in substituting for field experiments, which relaxes many constraints such as safety, cost, and even ethical concerns. These simulations along with advances in deep learning and surrogate modeling methods found use in several computational sciences and engineering fields for discovery and design. These fields span engineering, biology, economics, and many other social sciences. For example, [29] introduced AtomNet, the first structure-based, deep convolutional neural network designed to predict the bioactivity of small molecules for drug discovery applications. The authors of [30] applied deep learning neural network approaches in the context of computational protein design. Along with the above successes, several articles were recently published to advocate the introduction of such methods in their respective fields. For example, [31] advocated the introduction and application of deep learning and multilayer neural networks to agent-based models in economics. The authors of [32] advocated for and proposed a path for using data-driven approaches and data mining to model human decision making processes in the context of agent-based models in social sciences.

1.3 Background on trucking automation

Heavy duty truck driving performance is details sensitive [16], [33]–[37]. A typical passenger car could achieve a fairly consistent braking distances while the braking distance of a truck varies significantly within a single trip; before and after hooking up to a trailer, before and after trailer loading, and distance could double as the brakes worms up during the trip. Model and control accuracy have been shown to be significant (and limiting) factors for precision driving maneuvers [10], [15]. Deep learning has been shown to improve modeling accuracy compared to state of the art classical models for passenger cars [9], [10]. However, heavy duty truck modelling and control literature using machine learning—such as deep learning and deep reinforcement learning—is still sparse. Heavy duty trucks are typically configured and tailor built to optimize to their expected mission requirements. Detailed underlying physics and internal states of trucks are configuration specific and often are different from the more exhaustively modeled variants (components) of passenger cars.

Physics-based modeling and control of heavy duty trucks. Physics-based modeling have historically been an essential step for the design of model-based vehicle control systems [38]. The classical vehicle modeling framework is supported by a highly developed theory; however, in practice, the process involves significant art in addition to the science. Within the control community, vehicle models are tailored for application requirements in an attempt to balance model accuracy (level of details) and model complexity. Most vehicles are complex and highly non-linear systems. The model building (tailoring) process involves identification of the relevant details, selection (if already modeled, or building new ones) of suitable models for the desired operating range and conditions, and derivation of suitable approximations for the desired accuracy and complexity levels.

This process becomes more challenging with the increase in vehicle complexity (passenger car vs heavy duty truck [14], [39]), maneuver precision (single vehicle vs platooning control [14], [15]), operating range (normal vs high performance driving and drifting [40]), and driving conditions (dry vs snow covered roads [41]). Although still demanding, forefront nonlinear control theory such as model predictive control (MPC) have more relaxed constraints on the complexity of the model compared to the earlier linear control theory. Simulation based control verification, on the other hand, where no analytical manipulation is required, allows for models of highest level of details to be used.

Specific systems such as Cooperative Adaptive Cruise Control (CACC) cascades vehicles at short following distances at freeway speeds. CACC maneuvers operate at the margin of controllability/reachability, and thus require high precision control allowing for only split seconds to correct control errors with disastrous failure consequences. For passenger cars, typically light-weight, responsive and equipped with sufficient drive (and braking) power under normal operating conditions, in [39], a stable longitudinal CACC string was successfully developed using simple linear transfer function car model. On the other hand, long haul heavy duty trucks have lower power-to-weight ratio than passenger vehicles, suffer from significant response delays, and from actuator saturation [14]. To develop stable longitudinal CACC string of heavy duty trucks, the authors of [15] modeled vehicle dynamics (aerodynamics and gravity), power-train drive dynamics (engine, torque converter, transmissions, shafts, and tires), and brake system dynamics (engine brake, transmission retarder, and service brake). To design controllers with analytical guarantees such as bounds on control error, they used high-level of detail approximations to simplify their over-all model, where for example, the engine is modeled only by the torque it generates, and the transmission is modeled by the set of torque scaling gear ratios. However, with more flexible control design tools, namely, using reinforcement learning to tune patch adaptive controllers, even for a passenger car, the authors of [42] favored highly detailed actuation, power-train (engine, transmission, steering, braking, drive-train, suspension, and tire), and rigid body dynamics models to develop longitudinal and lateral CACC strings of passenger cars. Instead of using high-level approximations, their model captured the internal dynamics of each component such as the torque generating combustion process inside the engine.

In practice, these models must be fitted to the dynamics of the specific vehicle the control is applied to. Accurate reverse engineering of a specific vehicle of interest is challenged by physical features and analytical parameters that are difficult to model or estimate, and more importantly, by manufacturer-proprietary details such as engine maps and gear shifting schedules, and recently, by software defined operating modes such as eco-driving and sport modes. Other challenges include the availability of data from hand-designed experiments and the potential need for an expert driver to execute these experiments.

Deep approaches for vehicular applications. Deep learning (and deep-RL) methods have reached advanced modeling (and control design) accuracy levels that are arguably sufficient for several practical accuracy-sensitive and accuracy-critical applications; however, the literature for vehicle dynamics modeling and control is still sparse.

The authors of [43] and [42] published some of the early works on the use of machine learning methods for vehicular (applications) dynamical modeling and control. Namely, the authors of [43] used reinforcement learning to design a controller to stabilize a model-helicopter at inverted hover point. They used stochastic linear regression to fit a physics-based reduced model to data collected from an expert pilot. For more complex nonlinear controls, [42] demonstrated the use of reinforcement learning to learn (tune) gains of classical patch adaptive controllers for cooperative simulated passenger car driving using detailed physics-based models.

A recent spark of interest in (deep learning) neural networks for vehicle modeling had emerged. For aerial vehicles, the authors of [44] used *feedforward neural networks* (FNN) to model the dynamics of a physical-model helicopter, and the authors of [45] proposed a hybrid model that fuses white-box physics-based models with black-box deep recurrent neural network based models to model physical-model quadrotor dynamics. The authors of [46] proposed a simple FNN model that is suitable for model based *linear-quadratic regulator* (LQR) and demonstrated improved physical-model quadrotor flight control performance.

For ground vehicles, the authors of [9] conducted a comparative analysis of different deep learning models for the modeling of a full size physical passenger car. The authors of [10] used FNN to model the dynamics of a full size physical passenger car and used it to demonstrate improved control of high performance maneuvers based on classical controllers. Modeling and control of longitudinal dynamics of heavy duty trucks using deep learning and deep reinforcement learning have been investigated in [47]–[49], which constitute the bases for the work published in this manuscript.

1.4 Contributions of the thesis

We breakdown the technical contributions of this dissertation as follows:

- This thesis develops, studies performance, and demonstrates a process for modeling and control of heavy duty trucks using deep learning and deep reinforcement learning.
  - A model for longitudinal dynamics of heavy duty trucks was developed using deep learning.
  - Cruise control and Cooperative Adaptive Cruise Control systems were developed using deep reinforcement learning.
  - A systematic process was developed to streamline data collection, model development, control system development, and validation.
  - Utility, extensions, and automation software was developed for simulation and field validation of deep reinforcement learning controllers.
  - Process, model, and control were demonstrated and evaluated in simulation and for two differently configured full size trucks.

- This thesis develops and implements a structure preserving numerical solver for non-local conservation laws and studies its application for multi-scale vehicular traffic modeling and actuation.
  - A discretization scheme for a general numerical solver for multi-dimensional non-local conservation laws was derived based on the characteristic method.
  - Vectorized representation of the most computationally intensive operations of the numerical solver was derived.
  - A general numerical solver algorithm was developed and implemented.
  - PDE Traffic flow models and coupled PDE-ODE vehicle and traffic flow actuation models were implemented for the characteristic solver.

1.5 Outline of the thesis

The rest of the thesis is organized as follows:

Chapter 2: Technical Background

This chapter introduces the background and the literature relevant to heavy duty truck automation. It introduces the heavy duty truck automation framework and gives an overview of models and controllers used to develop driver assistant systems such as cruise controllers, adaptive cruise controllers and cooperative adaptive cruise controllers. The chapter also introduces the high-fidelity simulation framework used in the reset of the work, and an engineering overview of a real full-size semi-automated truck platform. The chapter ends with an introduction to deep learning and deep reinforcement learning background relevant to the work presented in the rest of the manuscript.

Chapter 3: Microscopic multi-vehicle modeling and coordination using deep learning and deep reinforcement learning

Heavy duty truck mechanical configuration is often tailor designed and built for specific truck mission requirements. This renders the precise derivation of analytical dynamical models and controls for these trucks from first principles challenging and tedious. The derivation often requires several theoretical and applied areas of expertise to carry through. This chapter investigates deep learning and deep reinforcement learning as truck-configuration-agnostic longitudinal modeling and control approaches for heavy duty trucks. The process is applied to simulation and real-full size trucks for validation and experimental performance evaluation. With the approach of this chapter, (a) truck model is trained off-line, and (b) feedback control is also trained off-line. Trained feedback controllers are applied online both in simulation and field test for validation and transfer performance assessment.

Chapter 4: Vehicle actuation using deep reinforcement learning

Heavy duty truck control system building processes have historically been dependent on detailed mechanical configuration of target trucks. This chapter evaluates model free deep-RL continuous control learning, transfer, and robustness as a configuration agnostic strategy for control system development. For this purpose, deep-RL cruise control policies are developed and validated in simulation and field experiments using two differently configured trucks.

Chapter 5: Multi-scale vehicle and traffic flow modeling and actuation using mean-field models

Heavy duty trucks often operate within shared road infrastructure with background traffic. Mean-field limits can be used to develop multi-scale models of the truck and its interactions with the rest of the traffic on the road. These mean-field limits yield nonlocal conservation laws for which the numerical solver algorithm literature is still sparse. This chapter presents a characteristic-based numerical solver scheme. Numerical solutions to traffic flow models and Lagrangian traffic flow actuation are presented.

Chapter 2

Technical Background

2.1 Introduction

Heavy duty trucks are custom designed and built for specific mission requirements such as long haul, urban delivery, mixing and construction, and mining. Choice of internal mechanical components are constrained by factors such as operations and economics. For instance, operations in mountainous regions introduce constraints on component weight and introduce emphasis on brake effectiveness and economics. On the other hand, long haul operations on flat roads benefit from component choice for fuel efficiency.

This chapter presents a general background on the framework for modeling and control of heavy duty trucks. It presents an overview of the models and a selection of control systems relevant to this thesis. It then presents a high fidelity simulation framework and an engineering overview of full-size truck development for experimental studies. The chapter then introduces background on deep learning and deep reinforcement learning relevant to the work presented in this manuscript.

2.2 Truck modeling and control

This section presents the approximate physics-based truck model proposed by [14] and shown in Figure 2.1. This model emphasize details for longitudinal and powertrain dynamics to be suitable for heavy duty truck control applications; it is considered to be simple enough for tractable controller design while being detailed enough to reflect essential vehicle dynamics [14], [38]. The section then presents hierarchical controllers [38] based on the presented model for cruise control and cooperative adaptive cruise control. These models and controllers are presented with minor changes from their original formulation by their authors for simplification while introducing relevant concepts to truck automation and have been implemented for experimental validation in later chapters.

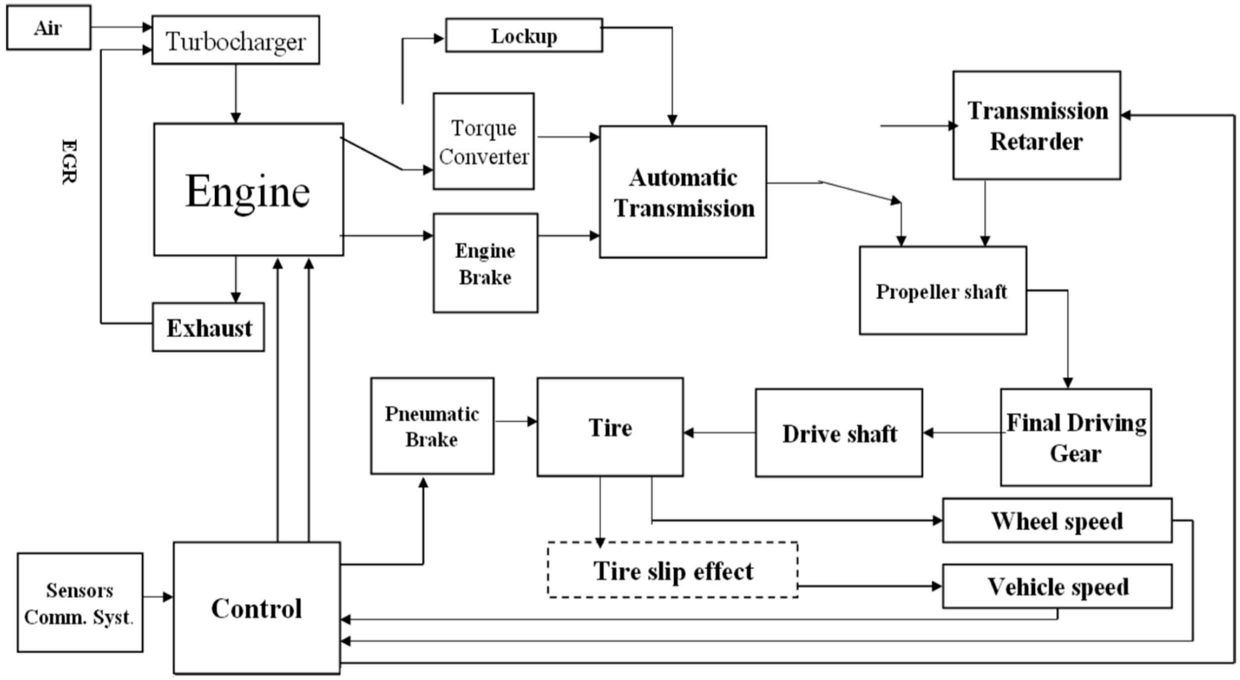


Figure 2.1: Model and control system block diagram for heavy duty truck longitudinal dynamics as presented in [14].

Vehicle modeling

This section presents a heavy duty truck longitudinal dynamics model that is approximated to suite analytical control design frameworks. The longitudinal motion of the truck is approximated by vehicle longitudinal dynamics, power-train dynamics, and brake system.

Vehicle dynamics model

The vehicle longitudinal dynamics model is derived in [14] based on Newton's method as shown in the following equation:

$$m\dot{v} = F_x - \frac{1}{2}A_f \rho C_d v^2 - C_r mg - mg\sin\theta,$$

where v is vehicle speed, F_x is traction force, m is truck mass, A_f is effective frontal area, ρ is air density, C_d is aerodynamics drag coefficient, Cr is rolling resistance coefficient, g is gravitational constant, and θ is road grade. In general, traction force is controlled only indirectly by commanding engine and brake torques, for which a power-train model, presented next, is essential.

Power-train engine drive model

The model presented here approximates the dynamics of wheels, gearbox, clutch, and engine as proposed by [14]. Wheel slip is assumed small and negligible for normal driving conditions. Hence, vehicle speed can be approximated by wheel speed multiplied by wheel radius as follows:

$$v = r_{\text{eff}}\omega_w,$$

where r_{eff} is effective radius and ω_w is wheel speed. The rotational dynamics of wheels and transmission shaft are approximated by the following equations:

$$I_w\dot{\omega}_w = T_w - F_x r_{\text{eff}},$$
$$I_t\dot{\omega}_t = T_t - r_G T_w,$$

where I_w is rotational inertia combining wheels, shaft connecting wheels, and gear box shaft that is closest to wheels; T_w is the torque on the wheel shaft; I_t is the rotational inertia of gear box shaft that is nearest to the engine; T_t is the torque on the shaft; and r_G is gear ratio.

The gear box model is approximated by its transformation of the rotational speed between its two sides according to the gear ratio as follows:

$$\omega_t r_G = \omega_w.$$

Gear shift schedule is typically software-defined and is manufacturer-propriety. Here it is approximated as function of engine throttle level and transmission shaft speed: gear shifts up when the shaft speed is increasing and the throttle level is decreasing, while gear shifts down when the shaft speed is decreasing and the throttle level is increasing.

A clutch, for manual transmission, or a torque converter, for automatic transmission, are used to decouple engine from gear box and allow for temporary rotational speed mismatch during gear shifts. Here, gear shifting is assumed fast and all associated disturbances are negligible. This assumption allows for approximating engine shaft torque and speed to be equivalent to transmission torque and speed as follows:

$$T_e = T_t,$$
$$\omega_e = \omega_t,$$

where T_e is engine output torque and ω_e is engine speed, which are command-able with an appropriate interface to some modern trucks.

Brake system model

Loaded long haul heavy duty trucks are significantly weighted vehicles requiring equally significant energy to brake. To balance cost, safety, and brake system requirements, these trucks are often custom built with a combination of service brake, engine retarder, and transmission retarder. Each one has a different general braking characteristics such as braking power limits, response function, response delay, and operating range.

Here, we omit presenting a detailed model (an example model can be found in [14]) because these are highly dependant on the specific setup installed on the truck. Instead, we briefly summarize the high-level model for service brake, for which we mainly need to consider wheel dynamics during braking as follows:

$$I_w \dot{\omega}_w = T_b - F_x r_{\text{eff}},$$

where T_b is the brake torque applied at the wheel. The brake torque is a nonlinear function of the brake cylinder pressure as follows:

$$T_b = f_b(P_b),$$

where P_b is the brake pressure at the brake cylinder and $f_b(\cdot)$ is a nonlinear function that depends on the design of the brake system.

Fuel consumption model

Fuel combustion in engine represents the primary energy consumed by the truck [50]. In addition to the models historically used for fuel consumption, fuel models can be detailed by incorporating explicit models to account for factors such as automation and CACC on fuel consumption [17].

Engine fuel consumption model. Approximate fuel consumption models can be derived based on engine combustion [50]. To simplify, engine is assumed to be operating at steady state while driving on freeway and engine transients are neglected. Steady state engine fuel consumption could be estimated using:

$$\text{Fuel Consumption [g/kWh]} = \frac{m_{\text{fuel}}}{\rho_{\text{fuel}} \cdot s},$$

where ρ_{fuel} is fuel density, s is distance traveled, and m_{fuel} is amount of fuel consumed given by:

$$m_{\text{fuel}} = \int_{t_o}^{t_{\text{end}}} \frac{\dot{m}_{\text{fuel}}}{3600} dt.$$

The flow of fuel to the engine $\dot{m}_{\text{fuel}}$ is given by:

$$\dot{m}_{\text{fuel}} = \omega_e \cdot \delta_{\text{e}} \cdot \frac{z_e}{2} \cdot \frac{60}{1000},$$

where ω_e is engine speed, δ_{e} is engine fueling [mg/stroke], and z_e is number of engine cylinders.

CACC fuel reduction effect model. Driving at short following distances as in CACC reduces drag and thus fuel consumption. Fuel reduction from CACC is sensitive to factors such as controller performance, vehicle geometry, driving distance, vehicle position within the CACC string, vehicle mass, and weather conditions [17], [51]–[54]. For a three-truck CACC string, individual truck fuel reduction model as a function of separation distance is shown in Figure 2.2 based on field experimental results [17]. This model suggests that all vehicles participating in the CACC string formation, including the leading vehicle, benefit from fuel reduction. The middle vehicle in a three-vehicle string can gain up to 17% reduction at about 4m separation, while the leader and trailing could gain up to 10% at the same separation distance. The gain for trailing vehicle peaks at about 13% reduction at 12m separation distance. In general, fuel reduction is inversely proportional to separation distance.

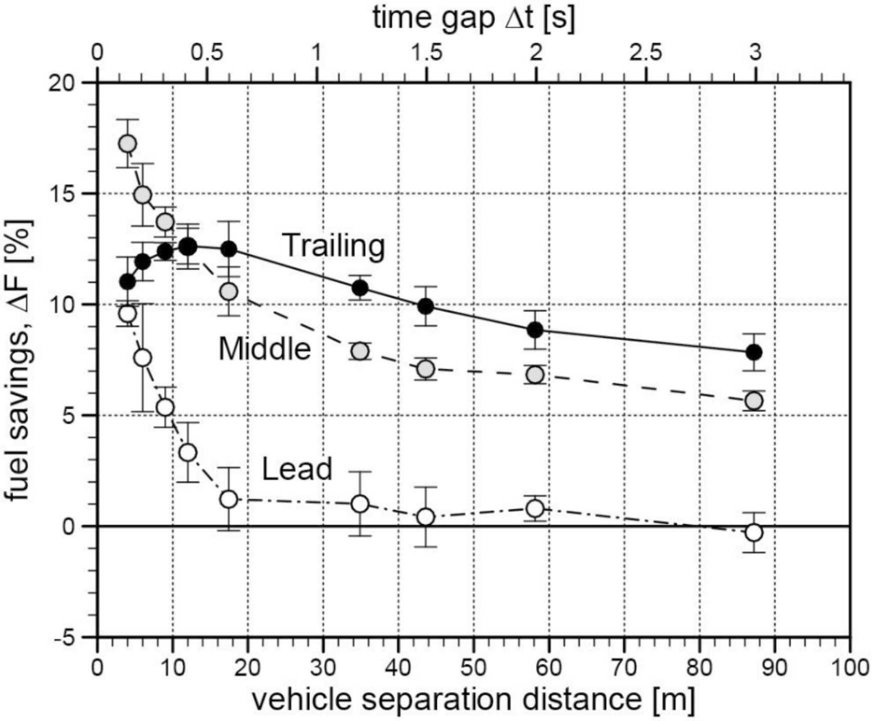


Figure 2.2: CACC fuel saving model from experimental results as presented in [17].

Discussion

This section presented an overview of the classical modeling framework along with the major models used for heavy duty trucks. More truck configuration specific details are often required for high precision modeling. Moreover, detailed actuation delay and saturation modeling and analysis are crucial for high performance and safe control.

Single vehicle automation: longitudinal cruise control

The foundational layer of heavy duty truck automation attempts to manage the complexity of the operation of a heavy duty truck. An experienced heavy duty truck driver manages the operating point of their truck such as speed and selected gear with careful consideration to truck load, surrounding traffic, and terrain to optimize for safety and operational factors such as wear and tear.

Heavy duty truck cruise controllers are designed for the basic function of truck speed regulation; predictive cruise controllers have been proposed to incorporate look ahead information and planning functions [33], [34]. Truck parameters such as mass, air drag, and rolling resistance may vary during its operation cycles; estimation and information systems for these parameters and road grade are often implemented to reduce sensitivity of truck to controls [35], [36]. The truck dynamics is sensitive to how it is operated during a trip introducing constraints for the safe and recommended operation procedures; for instance, trucks are often built with several brake systems providing the driver a menu of options varying in operability requirements and expected performance, sensitivity to operation, responsiveness, and cost of operation. Tailor designed controllers and optimizers are often employed to manage the operations of these systems, see for example the work presented in [37] and [16].

Cruise Control

This section presents a hierarchical controller as a baseline for longitudinal speed regulation of heavy duty trucks with engine torque and service brake actuation as suggested by [14]. This controller was implemented and used for validation in Chapter 4.

Hierarchical Controller. The controller considered in this section is a two-level feedback linearization hierarchical controller consisting of an upper level controller and a lower level controller [14], [38]. The lower level controller approximately linearizes (up to the level of modeled details) the dynamics of the system by mapping vehicle acceleration into the corresponding actuation control variables; engine command and brake command. This allows for the design of a simpler and more modular upper level controller assuming linear kinematics model.

Lower-level controller. The lower-level controller transforms desired acceleration into direct engine torque command $\hat{T}_e$ and/or brake pressure command $\hat{P}_b$ based on the inverse of the dynamical model presented in Section 2.2 as follows:

$$(\hat{T}_e, \hat{P}_b) = \begin{cases} (\bar{T}, 0) & \text{if } a_{\text{des}} > 0 \\ (0, \bar{P}_b) & \text{otherwise} \end{cases},$$

where

$$T_e = \frac{(m r_{\text{eff}}^2 r_G^2 + I_t + I_w r_G^2) a_{\text{des}}}{r_{\text{eff}} r_G} + \frac{1}{2} A_f \rho C_d v^2 r_{\text{eff}} r_G + C_r m g r_{\text{eff}} r_G,$$

$$P_b = f_b^{-1}(\frac{(mr_{\text{eff}}^2 + I_w)a_{\text{des}}}{r_{\text{eff}}} + \frac{1}{2}A_f\rho C_d v^2 r_{\text{eff}} + C_r m g r_{\text{eff}}),$$

and $f_b^{-1}(\cdot)$ is the inverse function of $f_b(\cdot)$.

Upper-level cruise controller. Assuming vehicle response is approximately linear given the lower-level controller, upper-level cruise controller (CC) could be designed based on the single integrator model $\dot{v}(t) = a(t)$ and error model $e_{CC}(t) = v_{\text{ref}}(t) - v(t)$, where t is time, $v_{\text{ref}}(t)$ is reference speed, and $e_{CC}(t)$ is CC error. Without detailed performance requirements, a simple P-controller could be used to design a stable cruise controller. In the P-control, desired acceleration, $a_{\text{des}}(t)$ is picked equal to $k_P(v_{\text{ref}}(t) - v(t))$, where k_P is the control gain tuned for stability and performance requirements such as ride comfort.

Adaptive Cruise Control and Cooperative Adaptive Cruise Control

Cooperative Adaptive Cruise Control (CACC) [55] is generally designed to stabilize a string of vehicles driving at short following distances at freeway speed as shown in Figure 2.3. The first vehicle is often assumed human driven or controlled using a cruise controller or an adaptive cruise controller [15], [56]. Several spacing policies have been considered such as constant following distance and constant following time-gap [57].

CACC variations have been implemented by several groups [58]–[61]. Emissions and fuel are two significant factors in trucking, which motivates the further development and study of CACC systems. Fuel reduction have been demonstrated in the field [17], [62]. Simulation studies also suggest potential for an increase in road flow capacity [18] proportional to technology penetration rate. Further theoretical and simulation studies noted a potential of CACC strings operating as moving bottlenecks potentially contributing to a drop in capacity [63], [64].

CACC is an extension to Adaptive Cruise Control (ACC) systems by the introduction of intervehicle communication [15]. Intervehicle communication replaces sensor measurements of speeds and distances to reduce feedback loop delays sited to cause string instability as demonstrated in simulation [65] and field experiments [66]. String stability [67] and practical string stability [68] characterize the propagation of control error in interconnected and cascade systems such as CACC strings. Compared to ACC, CACC enhances synchronization of maneuvers, which could lead to improved safety even with shorter following time gaps.

System performance is further optimized by the introduction of centralized coordination layers. Local vehicle string coordination has been studied to compensate for local road topography variations in [69]. Longer term coordination has been studied to plan CACC string formation along the route of participating trucks in [70]–[72].

Vehicle following controller

Stable vehicle following maneuver is the primary maneuver in CACC systems. The maneuver has been implemented using several control frameworks such as hierarchical control

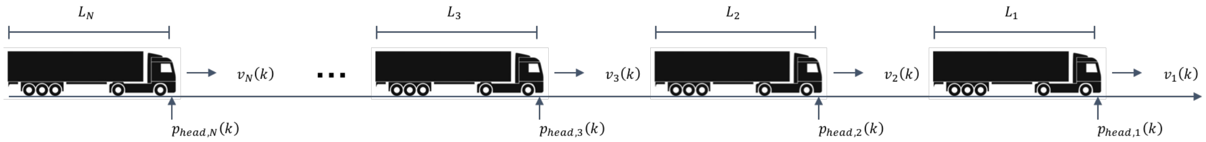


Figure 2.3: Heavy duty truck Cooperative Adaptive Cruise Control String.

[15], sliding mode control [73], and model predictive control [69]. This section presents a simplified hierarchical control scheme as a baseline based on the work presented in [15]. This hierarchical controller utilizes a lower-level controller as presented previously in Section 2.2 (Cruise Control). The upper level controller is designed to regulate time-gap and speed of a two-truck CACC string.

Upper-level controller. In a two truck cooperative adaptive cruise control (CACC), a controller is designed for the second truck (follower or ego) in the string. CACC is designed to simultaneously regulate following speed and following time gap relative to the first truck (leader) in the string. Assuming vehicle response is approximately linear given the lower-level controller, CACC could be designed assuming double integrator model:

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ a(t) \end{pmatrix},$$

where $x(t)$ is longitudinal position assuming $x(t)$ increases along vehicle forward direction. The corresponding error model hence becomes:

$$e(t) = \begin{pmatrix} e_x(t) \\ e_v(t) \\ e_a(t) \end{pmatrix} = \begin{pmatrix} x_{\text{leader}}(t) - x_{\text{ego}}(t) - L_{\text{leader}} - Tg \cdot v_{\mathbf{ego}}(t) \\ v_{\text{leader}}(t) - v_{\text{ego}}(t) \\ a_{\text{leader}}(t) - a_{\text{ego}}(t) \end{pmatrix},$$

where $e(t)$ is error vector, $e_x(t)$ is distance error, $e_v(t)$ is speed error, $e_a(t)$ is acceleration error, $x_{\text{leader}}(t)$ is leader longitudinal position, $x_{\text{ego}}(t)$ is ego longitudinal position, L_{leader} is leader vehicle length, Tg is desired time gap, $v_{\text{leader}}(t)$ is leader speed, $v_{\text{ego}}(t)$ is ego speed, $a_{\text{leader}}(t)$ is leader acceleration, and $a_{\text{ego}}(t) = a_{\text{des}}(t)$ is ego acceleration, which is the controllable variable. As shown in [15], the error function linear combination $e_a(t) + k_1 \cdot e_v(t) + k_2 \cdot e_x(t)$ can be stabilized to zero using the control law:

$$a_{\text{des}}(t) = -(k_1 + k_2 Tg) \cdot (v_{\text{ego}}(t) - v_{\text{leader}}(t)) - k_2 \cdot (x_{\text{ego}} - x_{\text{leader}}) - k_2 \cdot Tg \cdot v_{\text{leader}},$$

where k_1 and k_2 are control gains chosen such that $H(s) = s^2 + (k_1 + k_2 \cdot Tg) \cdot s + k_2$ is Hurwitz polynomial for $Tg > 0$ and $v_{\text{leader}} > 0$.

2.3 High fidelity simulation of trucks

TruckSIM [74] is a state-of-the-art commercial software framework for truck modeling, simulation, and analysis with high fidelity modeling capabilities and a detailed vehicle library as in Figure 2.4. The framework provides libraries of pre-modeled truck components spanning engines all the way to tires and chassis as in Figure 2.5. The framework also provides utilities to model new trucks using off-the-shelf customizable models from the TruckSIM library, or through user provided external-models implemented in frameworks such as MATLAB/Simulink. This simulation framework has been used throughout the remainder chapters of this thesis.

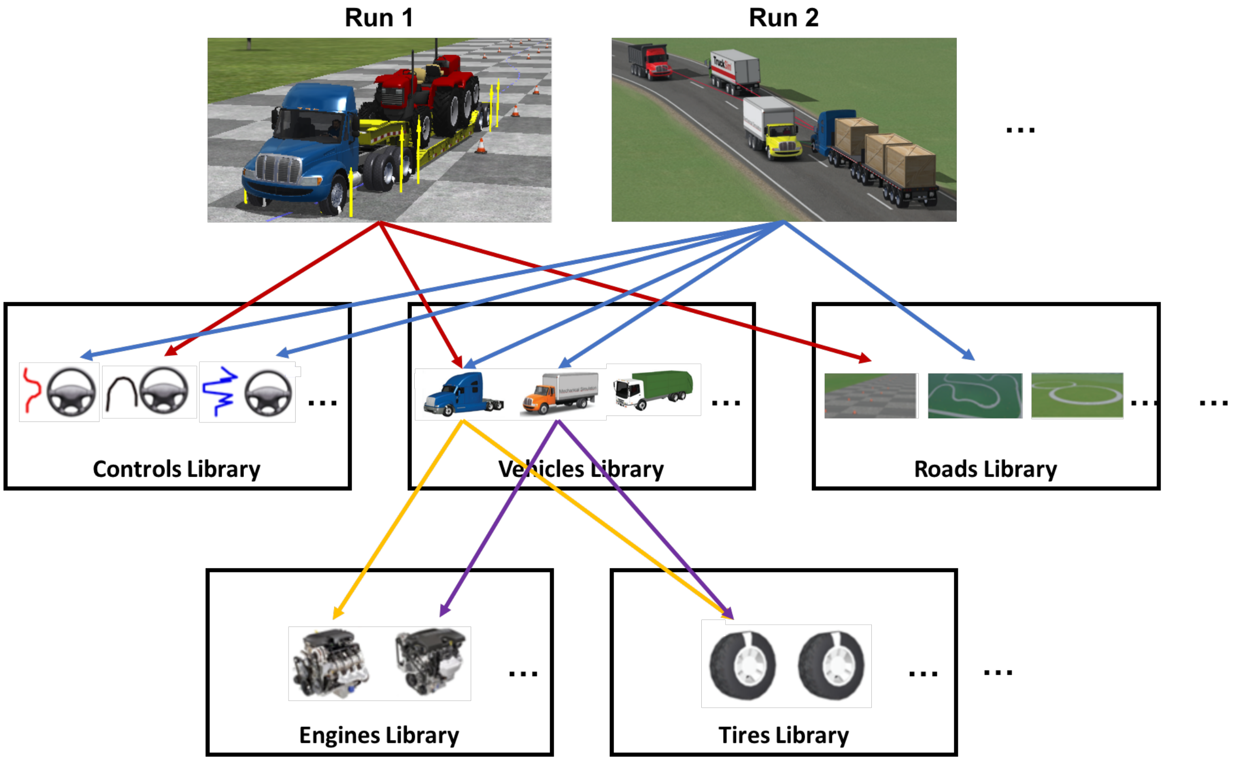


Figure 2.4: Truck models and vehicle component libraries of TruckSIM [75].

Modeling and simulation

TruckSIM provides a simulation engine for their highly detailed models. Each TruckSIM model is comprised of three primary sub-models as shown in Figure 2.5: vehicle model, a simulation procedure, and an external model interface. The vehicle model is a composition of highly detailed dynamical models of truck parts such as engines and tires. These models of parts encapsulate detailed models of the components within each part such as engine and transmission models within the powertrain part.

The simulation procedure specifies how and where the vehicle drives by means of driver controls and road specification. Each control, such as throttle control, can be specified as low-level control, for example, an engine torque, or more abstractly as an automated driver model, for example, a cruise control speed profile. The road and driving environment specifies road surface and environmental details such as road 3D geometry, friction, and wind.

The interface model specifies import and export variables to facilitate TruckSIM model extensibility and integration with other frameworks.

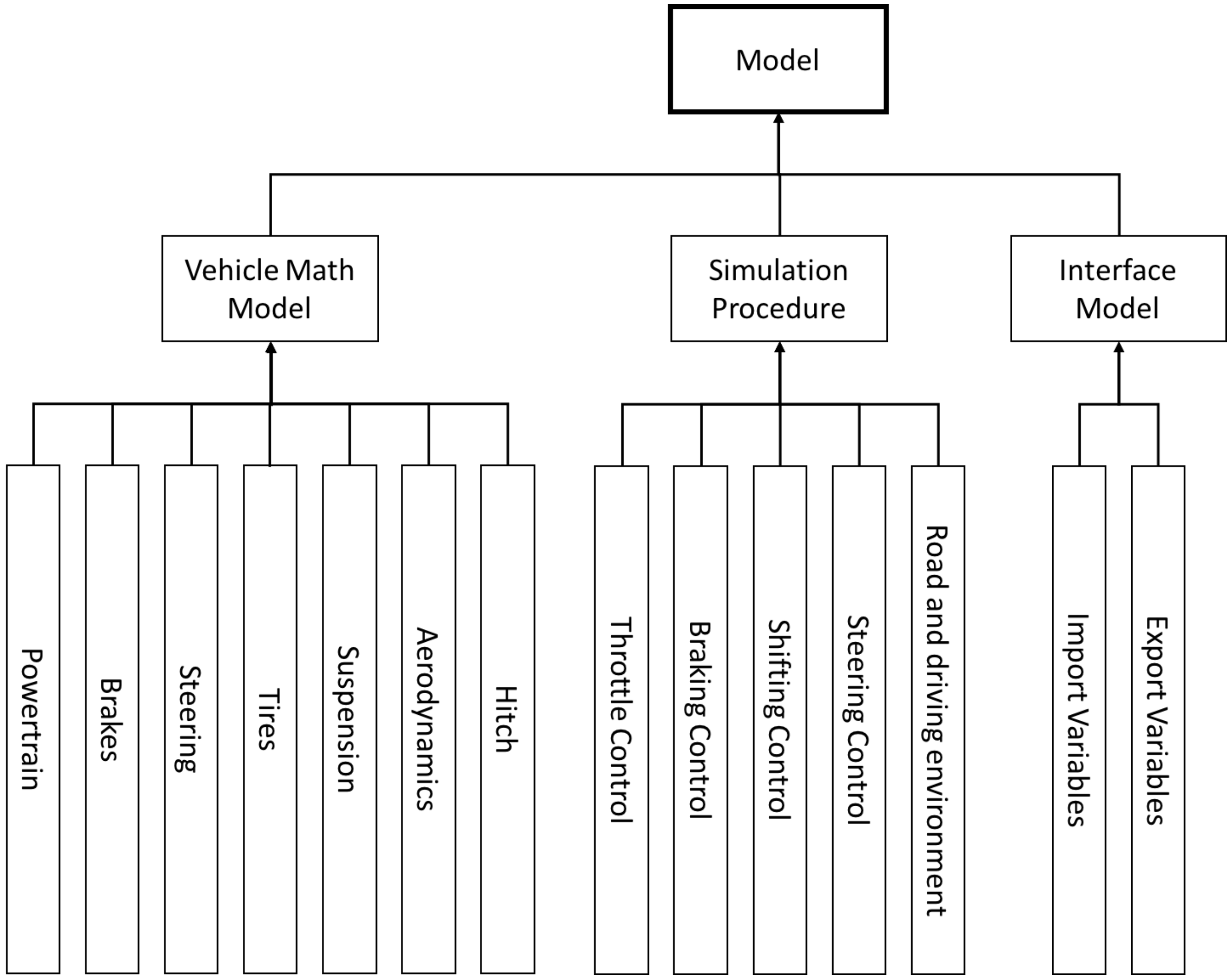


Figure 2.5: TruckSIM model decomposition.

Multiple-vehicle simulation scenarios

A multiple vehicle simulation scenario could be implemented by decomposing the scenario into a set of vehicles and a shared road and driving environment. TruckSIM framework provides a foundational coupling model, which can be extended in frameworks such as Simulink as in Figure 2.6. Simulink blocks of TruckSIM models implement the interface model shown in Figure 2.5. Additional sensors such as radars are provided to model sensor behavior and dynamics.

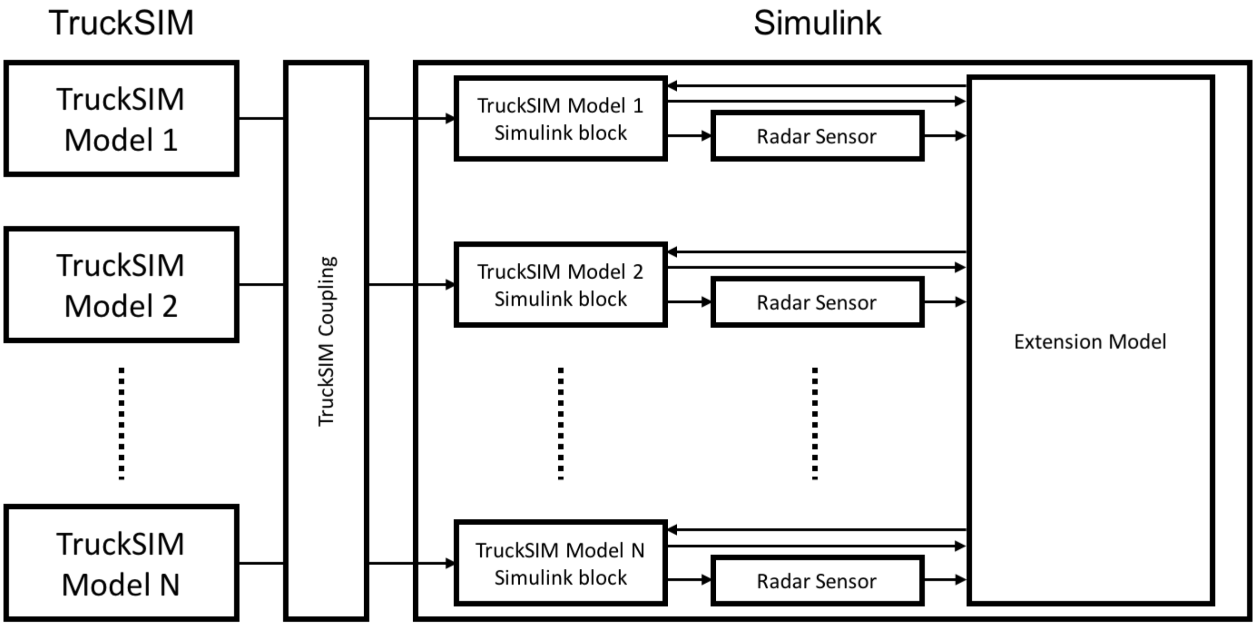


Figure 2.6: Architecture for TruckSIM model extension using Simulink.

Control system implementation

For this dissertation, control systems were implemented as TruckSIM model extensions in frameworks such as Simulink as in Figure 2.6 and in Python. Figure 2.7 shows control system implementation architecture for experimental evaluation. The Simulation Model encapsulates all TruckSIM models and sensors, and the Controller encapsulates the end-to-end (sensor to actuation) control system for all trucks in the system. The Scenario Generator encapsulates model and control scenario specifications along with generative models necessary for randomized scenario trials, while the Reset Model implements simulation (model and controls) management logic. The data collection layer provides the logic to manage and store simulated datasets.

2.4 Real full-size truck platform

An in-house custom-built truck automation platform is designed and implemented for the the primary purpose of field experimentation of feedback loop controls as in Figure 2.8 and open loop data collection Figure 2.9. This section provides a high level overview of system architecture and implementation of platform development.

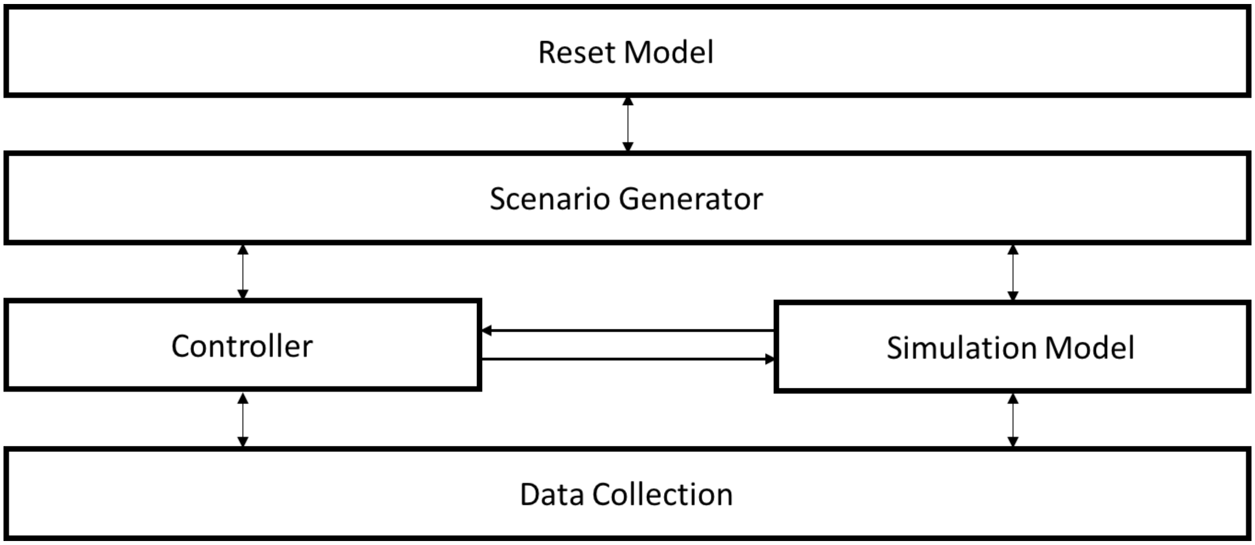


Figure 2.7: Architecture for experimental evaluation of truck control.

The feedback loop in Figure 2.8 is comprised of a control algorithm and the heavy duty truck as a mechatronic/robotic system. For convenience, truck mechatronic systems are divided into the mechanical systems and powertrain relavent to truck actuation and internal sensing and measurement systems. The primary actuators in the truck are engine drive, service brake, and engine brake. Internal sensing and measurement systems are built-in into the truck to provide internal truck state feedback. Additional sensors and communication systems are installed to provide sensing and communication with truck surrounding.

In addition to the feedback loop, the truck platform is used for driving data collection as in Figure 2.9. In open-loop data collection mode, the truck is commanded by a human driver, and the output of sensing and measurement systems are logged into local storage.

The automation platform is built for a Volvo FH16 as shown in Figure 2.10. This platform was developed at King Abdulaziz City for Science and Technology (KACST; Riyadh, Saudi Arabia) based on the previous platforms and expertise at California PATH (Richmond, California). For this dissertation, contributions included project and team development at KACST, co-leadership and coordination of the joint research and development team between KACST and PATH, coordination of team communication and travel, establishment of the cooperation with Zahid Tractor, organization and planning of a technical demonstration of the first truck automation project in Saudi, lab development and truck and equipment purchasing at KACST, and technical reverse engineering of truck mechatronic systems.

The following subsections provide an overview of the software and hardware subsystems used to implement these loops.

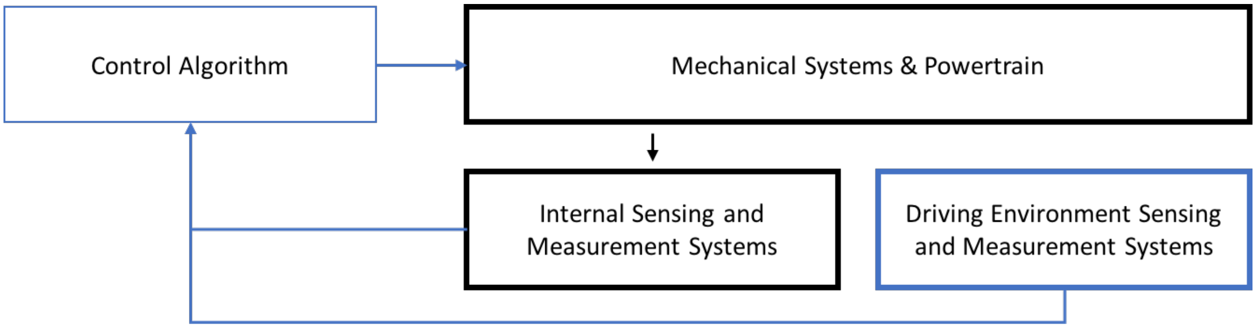

Control Algorithm
Mechanical Systems & Powertrain
Internal Sensing and Measurement Systems
Driving Environment Sensing and Measurement Systems

Figure 2.8: Feedback control loop architecture.

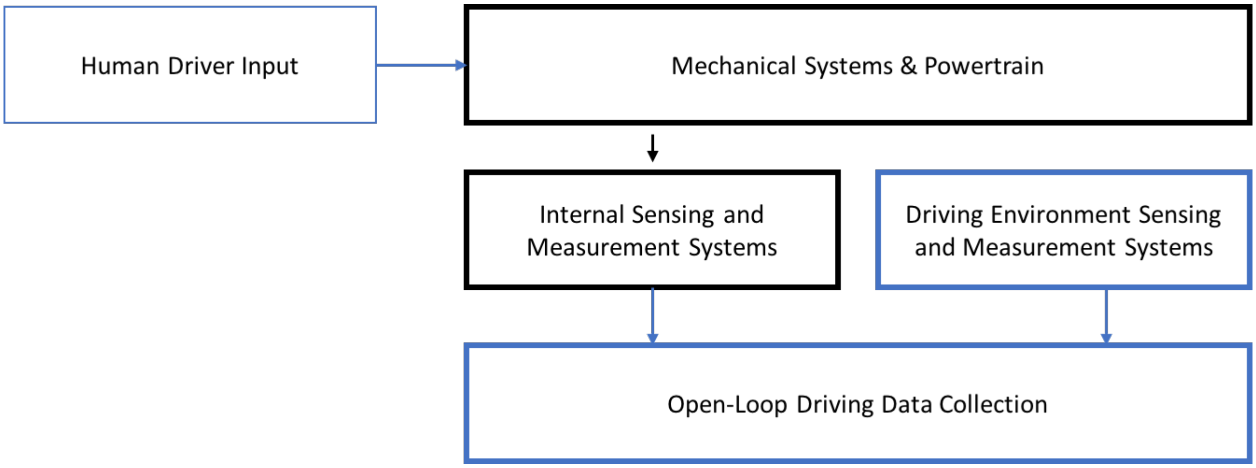

Human Driver Input
Mechanical Systems & Powertrain
Internal Sensing and Measurement Systems
Driving Environment Sensing and Measurement Systems
Open-Loop Driving Data Collection

Figure 2.9: Open loop data collection architecture.

Automated Driver Kit Hardware

The Automated Driver Kit (ADK) hardware shown in Figure 2.11 consists primarily of an Automated Driver Control Unit (ADCU), Global Positioning System (GPS), Radar/LiDAR system, Dedicated Short Range Communications (DSRC) On-Board Unit (OBU), and Hardware Failsafe system.

The ADCU interfaces directly with the driving environment sensing and measurement system; GPS, Radar/LiDAR, and DSRC; and with the truck's power and mechatronic systems. The ADK is custom designed to tolerate the physical storage and driving environment of the truck. The ADCU is installed inside the driver cabin for ease of access and development purposes, while GPS and DSRC antennas and Radar/LiDAR scanners installed outside of the cabin.

Custom-built interfaces are built to connect the ADCU with truck's mechatronic (for actuation and access to internal sensing and measurement systems) and power systems. The

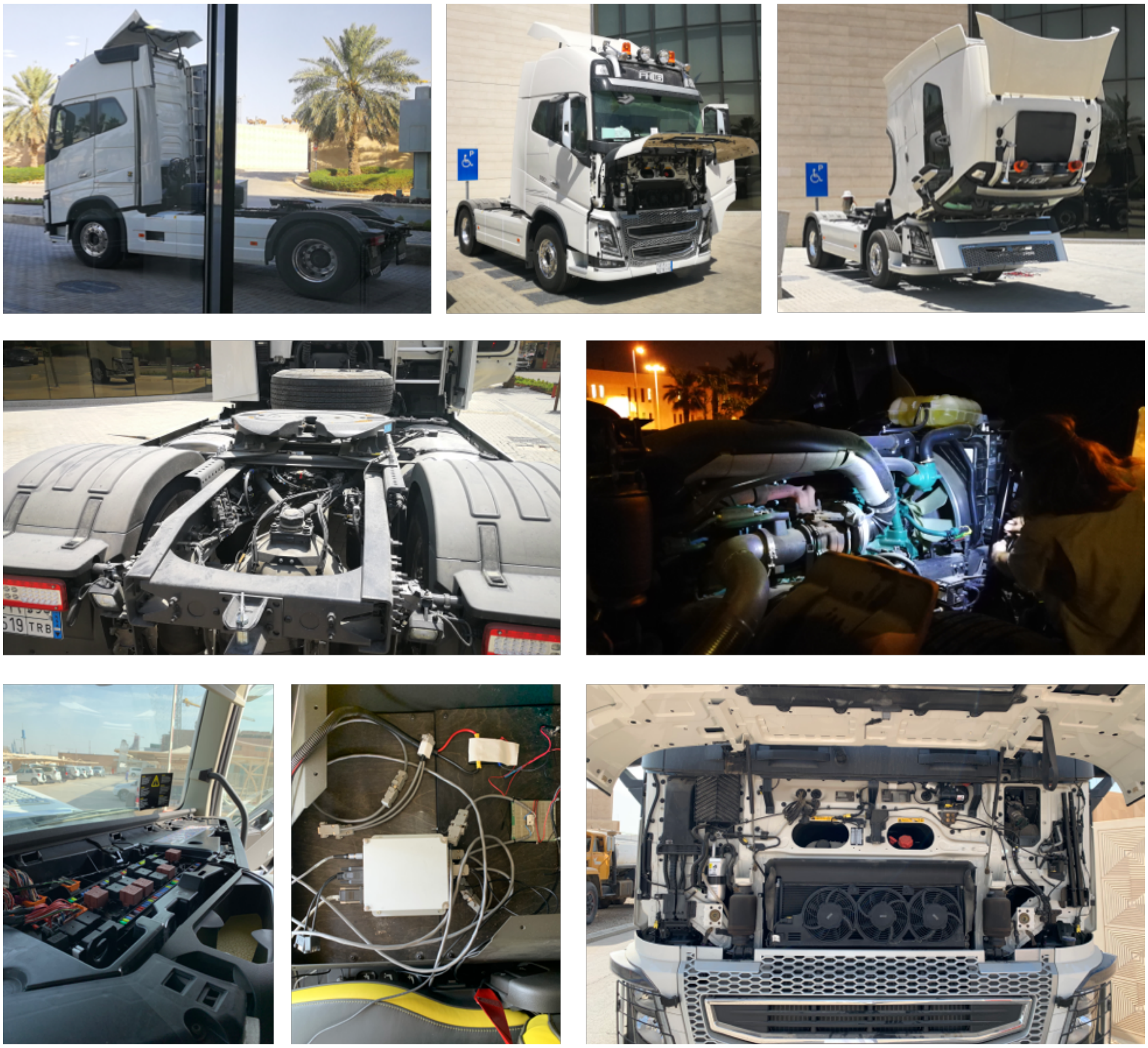


Figure 2.10: Heavy duty truck platform while still under development.

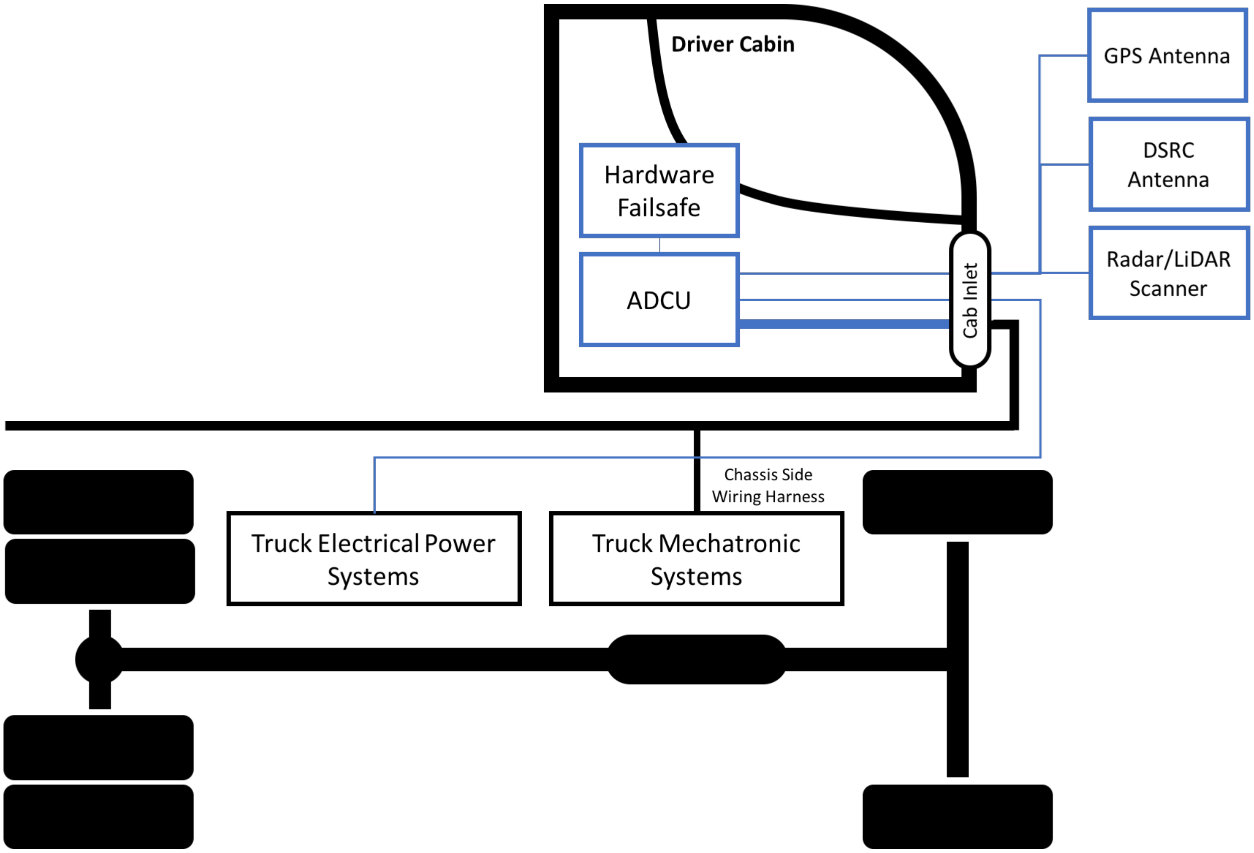


Figure 2.11: Hardware architecture for truck automation platform.

ADCU is integrated with truck's backbone used for communication between its subsystems to interface with the mechatronic system. A hardware failsafe system is implemented at this interface to terminate automatic control and return control to the driver. The system is powered directly from truck's main batteries.

The following subsections briefly describe each component.

ADCU. The ADCU is implemented using PCIe-104 standard; a small factor industrial grade standard for embedded control systems. The PCIe-104 CPU board, representing the main control computer (the main unit of the PCIe-104 system), interfaces with all other subsystems and executes the real-time control algorithms. The PCIe-104 is mounted inside the cap of the truck for easy access during development and testing.

Power to ADCU. The PC-104 receives its power through a 24Vdc-to-5Vdc power supply, which feeds from the power lines connected directly to truck's main batteries.

J-Bus Hardware Interfaces. J-1939 (J-bus) is a communication standard used in heavy duty machinery such as trucks. The J-bus uses CAN-bus standard as its physical layer, which is used as a communication network between vehicle subsystems. Multiple networks may exist inside the same vehicle, each of which dedicated for specific functions/subsystems (critical functions such as engine and brakes, assistance functions such as sensors and targeting, and other general purpose vehicle information), and are separated for reasons such as safety. The PCIe-104 CPU board interfaces with the J-buses of the vehicle using PCIe-104 CAN cards. Each CAN card makes available a single or multiple channels to interface with different networks of the vehicle. Each physical channel is bi-directional; however, actual read and write access privileges are constrained by manufacturer/assembler of the vehicle and its subsystems.

Hardware Failsafe Emergency Switch. The emergency switch connects and operates on the hardware directly and functions as a safety mechanism. It allows for guaranteed instantaneous disconnection of the automation system and returns full control back to the driver. This switch can be used to mitigate any effects the failure of the system may cause, and to rapidly contain unexpected behavior.

GPS. A high frequency and high precision GPS units are used for precise positioning of the vehicle, and also used as a global clock for synchronization functions, such as those necessary for proper communication between vehicles. GPS interfaces with the PCIe-104 through USB or RS-232 serial port.

DSRC. Dedicated Short Range Communications (DSRC) is short range communication system standardized for use in intelligent transportation systems (ITS) applications. Among the many configurations in which it can be used, this project focuses on using On-Board-Units (OBU) for reliable vehicle-to-vehicle (V2V) communication. The OBU unit interfaces with the PCIe-104 CPU board through RS-232 serial port. To communicate with other trucks, the OBU transceiver is connected to two diversity antennas connected at the two sides of the truck to allow for maximum reception at all times.

Radar/LiDAR. A Radar/LiDAR combination is used as primary real-time sensors for surrounding vehicles and other obstacles on the road. Radar systems are reliable for speed measurement of moving objects and LiDAR is reliable for targeting and distance measurement of surrounding objects. For longitudinal control systems, frontal Radar/Lidar combination is used.

Other Interfaces and Functions. Other interfaces and functions are made available for development, troubleshooting/debugging, and logging purposes. This include ethernet, wifi, low-level General Purpose IO, hard-drives, etc.

Automated Driver Kit Software

The Automated Driver Kit software is hosted the PCIe-104. The software consists of main operating system, the control algorithm, ADK subsystem drivers and interfaces, process coordinator, inter-process communication, and software failsafe system. The software coordinates the interconnection between all the subsystems and executes the feedback cooperative control law. Figure 2.12 highlights the most relevant components and illustrates the interconnection between them. The following subsections briefly describe each component.

RTOS. PCIe-104 embedded system run a Real-Time Operating-System (RTOS) for guaranteed deterministic performance and safe operation of the vehicle. The RTOS provides and

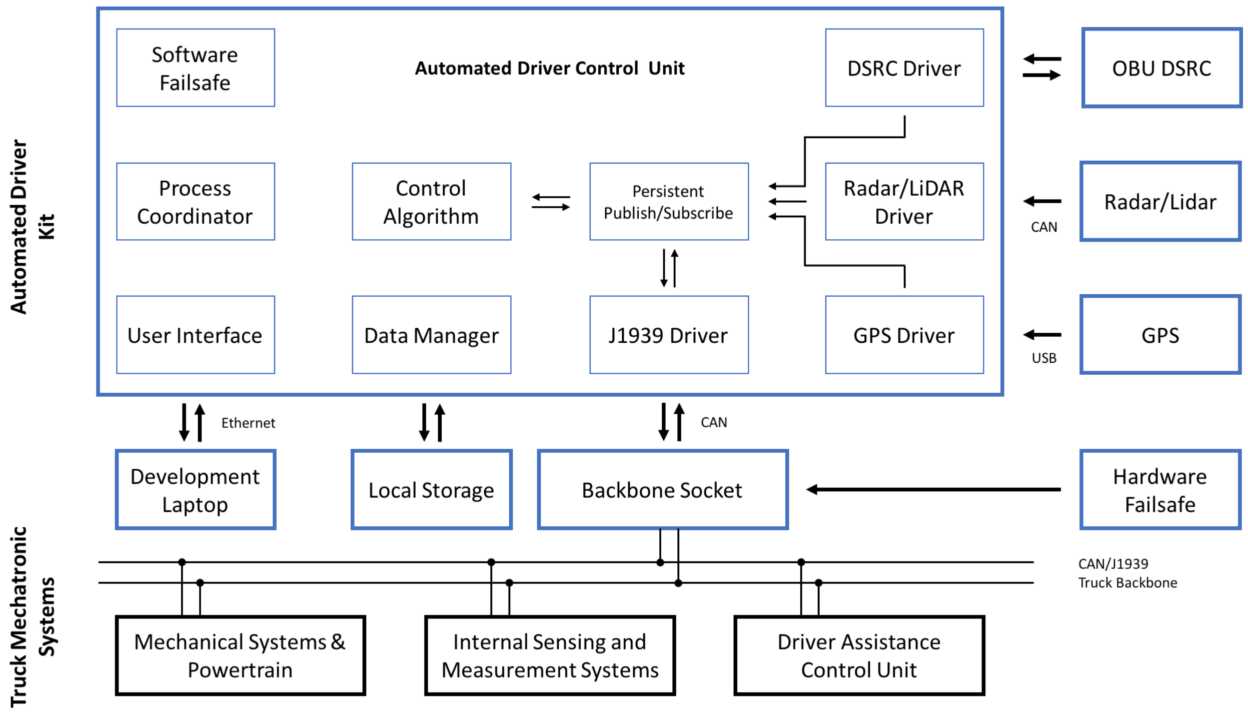


Figure 2.12: Software architecture for truck automation platform.

coordinates access to the the hardware layers necessary for operation, allows for the development and execution of high level software and file systems, and helps with scheduling and communication between processes. QNX Neutrino® RTOS is a commercial grade Unix-like operating system developed and currently maintained by Blackberry.

Real-Time Control Database. The Real-Time Control Database is implemented as a Persistent Publish/Subscribe (PPS) service. PPS abstracts access-coordination and data-sharing. It facilitates communication and data-sharing between processes; allowing for a flexible and robust loosely coupled complex real-time software system. The PPS service runs as a singleton process inside the PC-104 embedded system and is accessible by all other processes. Each process is provided with read-write privileges to create, update, and read real-time variables.

J-Bus Software Interface. The J-Bus software interface consist of low-level device drivers, high-level signal handlers, J-1939-Standard to SI-Standard (or an equivalent high level format) translators, and an interface with the Real-Time Control Database. This interface provides read-write access to the available CAN-networks in the vehicle. Read (from vehicle) functions maintain up to date information about the truck and J-bus-shared sensor information inside the database. Write (to the vehicle) functions are used to execute commands requested for the automation of the truck (by the control algorithm). J-1939 bus messages are low-level messages that are encoded and packetized to stream through a communication network. The high-level signal handler is the closest layer (next to the driver) to hardware to operate on J-bus data. It coordinates encoding, decoding, packetizing, and unpacking messages when they are received and before they are transmitted/redistributed on each side (PC-104 side, and vehicle side). This involves translation between J-1939-Standard encoding and units, and between the SI-system (or an equivalent high level format) used by the subsystems inside the PC-104.

GPS Driver. GPS software is primarily developed to read GPS position and global time information and make it available to the reset of the system. It uses RS-232 interface, which does not require tailored low-level driver. The GPS handler translates the global positioning and timing messages to formats suitable for local control (ex: spherical to cartesian coordinate systems).

DSRC Driver. Vehicle-to-Vehicle communication follows a standard communication protocol. DSRC uses RS-232 interface, which does not require tailored low-level driver. However, it requires the implementation of a software communication stack for encoding/decoding, serialization/deserialization, and packetization/unpacking. Moreover, to ensure reliable real-time operation, the software stack implements protocols to maintain reliable quality of service (minimize delay, packet dropping, etc). DSRC communicates safety messages and essential information for control system operation and for reliable real-time feedforward control functions.

Control Algorithms. The control algorithm aggregates all sensor and communicated information to execute a feedback control loop. For a cooperative adaptive cruise control system (CACC) for example, it consumes information such as positions, speeds, and accelerations of all vehicles in the CACC formation and detects cut-ins. It uses this information to compute new commands the engine or brake must execute at each control cycle. Engine and brake commands are the lowest low-level commands made accessible by the vehicle manufacturer.

Data logging. A logger runs as an independent process inside the PCIe-104 embedded system. It captures the state of the Real-Time Control Database at adjustable sampling intervals. The data is then stored either inside the PCIe-104 computer or in an external hard drive for off-line analysis. This is done via the built-in QNX database structure QDB.

Wi-Fi, Ethernet, and Laptop-Access. General purpose access to the PCIe-104 and its QNX operating system is facilitated by standard networking and network access software; namely, standard ethernet/wifi communication stacks, and SSH protocol.

2.5 Deep learning

Recent progress in machine learning, optimization methods, and sophisticated software abstraction techniques gave rise to deep learning. Deep learning methods are used to fit large and complex models to data. *Artificial neural network* (ANN) model for instance is one of the common parametric universal nonlinear function approximators [76]; theoretically expressive enough to approximate any finite dimensional function. Besides the ability of deep learning to fit large and complex models, these methods are capable of utilizing raw data, and hence avoid manual feature (problem variables) engineering and feature selection.

Advanced neural network architectures have been successfully trained using deep learning to solve (and stack and transfer solutions for) many practical problems. These architectures include *convolutional neural networks* (CNNs) for predictive image processing and understanding, *recurrent neural networks* (RNN) for language modeling and translation, and generative models for raw audio signal generation [77], [78]. Many of the optimized solutions outperformed hand-engineered ones, and more importantly, many of them outperformed or were on-par to human performance on the same task (where applicable).

Artificial Neural Networks

Artificial neural networks, feedforward neural networks, or simply neural networks are used synonymously in this chapter. Neural networks are parametric (with respect to their weights and biases) universal function approximators [76]. They are structured in layers that transform inputs to outputs [79]. The output of each layer is a linear combination of its inputs followed by a nonlinear activation function such as the sigmoid function. Outputs from each layer represent input to the following layer.

For each multidimensional (with respect to number of inputs and outputs) data point, this is represented algebraically as $o_i := \varphi_i(W_i o_{i-1} + b_i)$ for $i \in \{1 \cdots L\}$ where L is the number of network layers, W_i and b_i are weights and biases of layer i of layer respective size, o_i is output of layer i, $o_0 := x$ for $x \in \mathbb{R}^n$ is network input vector of n features, and $o_L \in \mathbb{R}^m$ is network output vector of m targets. φ_i is the non-linear activation function at layer i.

The nonlinear activation function, number of layers, and layer sizes are design choices (hyper-parameters) which affect the expressive power of the network. These hyperparameter, along with input-output scaling strategy, parameter initialization strategy, and choice of optimization strategy impact trainability and training speed for the network. ANNs are trained by optimizing an objective function such as $MSE = \frac{1}{k}\sum_{j=1}^{k} \|o_L^{(j)} - y^{(j)}\|_2^2$ for continuous regression over the parameter space of the network, namely, W_i and b_i, where $y^{(j)}$ and $o_L^{(j)}$ are, respectively, the ground truth output and the ANN prediction of the jth multidimensional data point.

Recurrent Neural Networks

RNN architecture is an advanced variant to ANNs that is used for time-series modeling. It was popularized for its ability to capture time dependencies and hidden (latent) state dynamics. In its most standard form, RNNs use output feedback (acting as memory cells) to propagate information through time. RNNs are trained by unfolding the network across time-steps and sharing a single set of parameters across all of the steps. However, standard RNNs suffer from vanishing/exploding gradients because, during training (optimization),

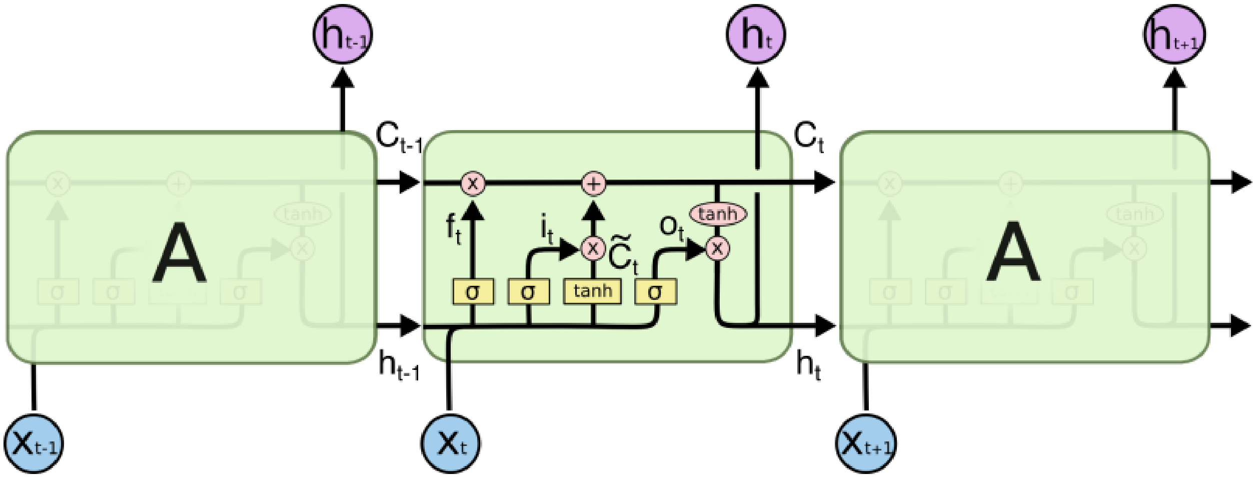


Figure 2.13: LSTM cells utilize internal memory variable C_t in addition to output feedbacks h_t to efficiently learn latent state dynamics and long term dependencies. [1]

derivatives get repeatedly multiplied across time-steps. This affects trainability (slow or stopping of optimization convergence). More sophisticated RNN architectures are used in practice to avoid these problems. Namely, in this manuscript, we use LSTM cells [80] shown in Figure 2.13. LSTMs utilize carefully designed gating structure to transmit information through an additional internal memory variable C_t, besides output feedback h_t, bypassing the repeated multiplications of the gradient.

Algebraically, LSTM output equation is $h_t = o_t * \tanh(C_t)$, where $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$ and $*$ is the element-wise vector multiplication operator. LSTM internal memory cell update equation is $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$, where $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$, $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$, and $\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$; tanh and σ are non-linear activation functions, respectively, the hyperbolic tangent and sigmoid functions; $x_t \in \mathbb{R}^n$ is input vector at time-step t, $h_t \in \mathbb{R}^m$ is output vector at time-step t, and $C_t \in \mathbb{R}^m$ is internal memory vector at time-step t; W_o, W_f, W_i, W_C, b_o, b_f, b_i, b_C are shared network parameters (neural weight matrices and neural bias vectors) of proper sizes; and f_t, i_t, $\tilde{C}_t$, and o_t are intermediate variables for presentation purposes.

LSTM networks are often trained using a truncated version of the unfolded network, and applying stochastic gradient methods (batch training). For example, the training set is split into k time-series examples of length l each. The weights and biases are trained iterative using batches of $\tilde{k}$ ($\leq k$) examples, where k, $\tilde{k}$, and l are hyperparameters. For continuous regression problems, a typical objective function to train the network is $MSE = \frac{1}{\tilde{k}l}\sum_{j=1}^{\tilde{k}}\sum_{t=1}^{l}\|h_t^{(j)} - y_t^{(j)}\|_2^2$, where $h_t^{(j)}$ and $y_t^{(j)}$ are, respective, the ground truth output vector and the LSTM prediction vector of the jth time-series example at time-step t.

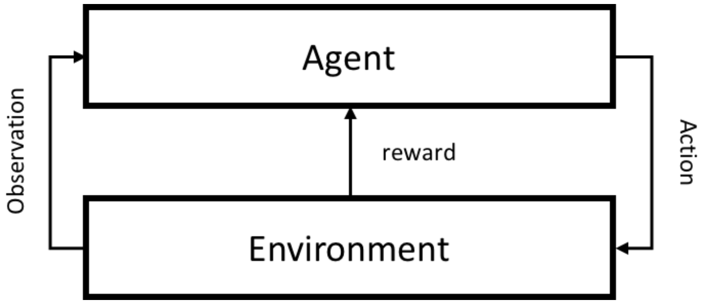


Figure 2.14: A simplified reinforcement learning system diagram.

Implementation and practical considerations

Tensorflow [81] is an open-source computational software library popular for implementation of deep learning algorithms. Models are described as computational graphs where nodes represent operations and edges represent data flows. Problem data are represented as placeholders for reusability and model parameters are represented as variables that can be fitted or estimated from data. Tensorflow library makes available several optimization algorithms for fitting model to data. Tensorflow optimization algorithms are iterative algorithms that utilize built-in automated differentiation of objective function with respect to relevant graph variables. A fitted model is portable and can be reused by feeding new data samples through model placeholders to evaluate respective outputs.

In practice, learning can be made efficient by applying standard pre-processing steps to clean and prepare data. Depending on specific model architecture, these steps include data alignment, handling of missing data, and data normalization or standardization. Additional practical considerations include choice of parameter initialization and hyper-parameter optimization method.

2.6 Deep reinforcement learning

Deep reinforcement learning (deep-RL) provides a formalism for learning of sequential decision problems with feedback including continues feedback control. Deep-RL problems are formulated with an environment and an agent. The environment abstracts the “world” the agent observes and act upon within the objective of maximizing its lifetime reward as shown in Figure 2.14. Deep-RL has been successfully used to solve many challenging problems such as the development of AI that learns by experience to play and win Atari games [82], Go [83], and others. It has been applied to continuous control problems [84] and continuous control problems in traffic control [85].

[1] This image and organized equations are courtesy of http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Design of continuous control problems can be formulated as a deep-RL problem modeled as a POMDP defined by the tuple $(S, P, OS, OP, A, r, \rho_o, \gamma, T)$, Where S represents the state space of an RL-environment (system); P is state transition probability space (governing dynamics of the system); OS represents the observable state space (space of system outputs); OP is probability distribution of observation space (governing dynamics of observation model); A represents the action space (actuation and control variables) of an RL-agent (a decision function, a policy, or a controller); r is reward function (system performance metric); ρ_o is initial state distribution; γ is reward discount factor over time; and T is time horizon. In this manuscript, we use model free Policy Gradient learning [84], [86] to computationally optimize for expected discounted cumulative reward for an agent policy π_θ parameterized by θ:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{t=1}^{T} E_{(s_t,a_t)\sim p_\theta(s_t,a_t)} \left[r(s_t, a_t) \right]$$

where s_t and a_t are state and action at time step t, and p_θ is probability distribution over state and action space.

RL-Lab [84] is a practical python framework for deep-RL. It provides interfaces to specify application details such as environment and agent models, initial distributions, and reward computation. It automates the learning process including the reward sampling and agent model optimization steps.

2.7 Conclusion

This chapter presented the standard framework used for heavy duty truck modeling and control and presented a high level background to the class of standard models and a selection of control systems. The chapter then presented fidelity simulation along with a high level background on software framework for simulation-based study and development of truck systems. The chapter concludes with a high level overview for the development of a full-size truck automation platform for field experiments. Next chapter uses this background to develop and demonstrate a modeling and control process for heavy duty trucks using deep learning and deep reinforcement learning.

Chapter 3

Microscopic multi-vehicle modeling and coordination using deep learning and deep reinforcement learning

3.1 Introduction

In this chapter, we develop a deep-learning-based longitudinal model for heavy duty trucks and validate its modeling accuracy for heavy duty trucks of different configurations both in simulation and using real-physical trucks.

Model-free deep reinforcement learning has been shown to achieve improved performance in many applications in addition to simplifying several previously intractable problems. Transfer of learned policies from simulation is often challenged however by the reality-gap (the mismatch between model and corresponding real-physical system). This chapter studies the application of deep learning for longitudinal modeling of heavy duty trucks and its application to minimize reality-gap for transferable deep reinforcement learning continuous control policies as shown in Figure 3.1.

The process uses deep learning to build deep replica models for each truck from some real vehicle pool. These deep replica models are used to develop deep environments suitable for deep reinforcement learning continuous control tasks. The chapter takes into consideration several of the factors either traditionally or expected to impact modeling and control performance such as vehicle mechanical configuration, operational scope, setup, and traffic scenarios.

Deep learning and deep reinforcement learning offer potential for improved performance at the expense of guarantees such as bounds on control error that are better understood using classical methods. To compensate, more in depth evaluation is always required. To simplify investigation and avoid expulsion of combinatorics however, the chapter focuses on presenting the process and experimental evaluation of the relevant components and leave additional investigations such as robustness for later chapters.

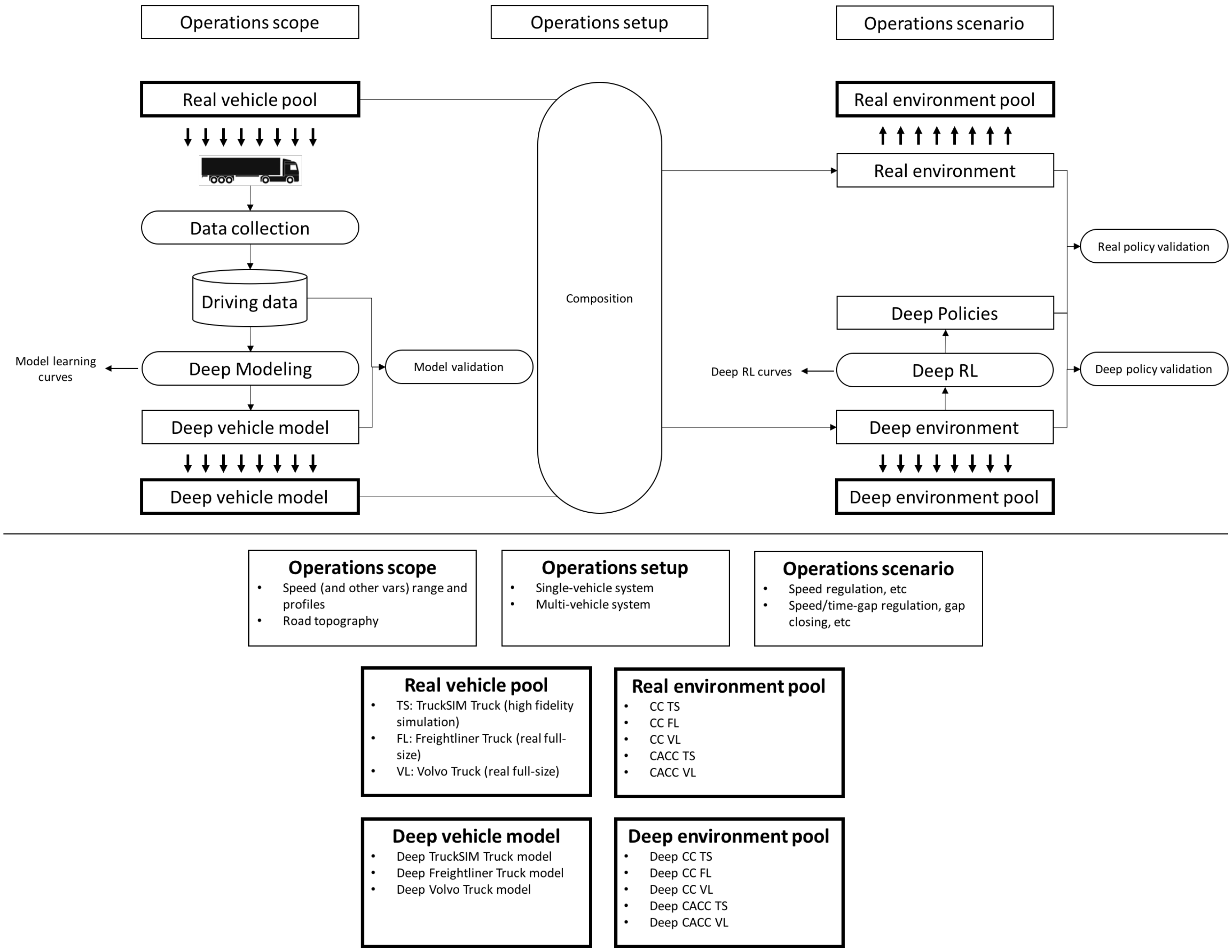


Figure 3.1: The deep truck process for the development of field-testable deep RL continuous control policies for longitudinal automation of heavy duty trucks and a sample of the pools and operational variations relevant to this work.

3.2 Modeling problem formulation

In this chapter, we formulate heavy duty truck longitudinal dynamics modeling as a time-series supervised deep learning problem. The longitudinal dynamics model f_{DT}, detailed in the next section, is represented as:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \end{bmatrix} = f_{DT}\left(\begin{bmatrix} u(k) \\ w(k) \end{bmatrix} \middle\| \begin{bmatrix} x(k) \\ y(k) \end{bmatrix}, \Phi \right), \tag{3.1}$$

where x represent internal state, y represent truck response, u represent controllable inputs to the truck, w represent uncontrollable conditions relevant to the dynamics, and Φ represent model parameters. The initial conditions are given by $x(k=0) = x_o$ and $y(k=0) = y_o$.

Model parameters Φ are trained by solving the following optimization problem:

$$\min_{\Phi} \quad \sum_{k} \|\hat{y}(k|\Phi) - y(k)\|_2^2$$

where $\hat{y}$ is the model-based estimate of y given ground truth historical driving time-series data y, u, w, initial state vector $x(k=0)$, and proper recursive substitution of the estimate of the internal state vector $\hat{x}(k|\Phi)$ for x as follows:

$$\begin{bmatrix} \hat{x}(k+1|\Phi) \\ \hat{y}(k+1|\Phi) \end{bmatrix} = f_{DT}\left(\begin{bmatrix} u(k) \\ w(k) \end{bmatrix} \middle\| \begin{bmatrix} \hat{x}(k|\Phi) \\ y(k) \end{bmatrix}, \Phi \right). \tag{3.2}$$

The model is trained using a K-step unfolded time-series mini-batch Adagrad (Adaptive stochastic gradient) algorithm [87], [88]. Each gradient step is estimated from M independent samples (time-series model evaluations) each of K time steps as follows:

$$\sum_{m=0\ldots M} \sum_{k=0\ldots K} \|\hat{y}_{n,m}(k|\Phi) - y_{n,m}(k)\|_2^2,$$

where the sub-indices abstract time-series splits and $n = 0\ldots N$ represent the mini-batch index. Training is initialized with random deep network parameters, and with $\hat{x}_{n,m}(k = 0|\Phi) = 0$ for all n and m. Given the trained model, truck simulations are generated from:

$$\begin{bmatrix} \hat{x}(k+1|\Phi) \\ \hat{y}(k+1|\Phi) \end{bmatrix} = f_{DT}\left(\begin{bmatrix} u(k) \\ w(k) \end{bmatrix} \middle\| \begin{bmatrix} \hat{x}(k|\Phi) \\ \hat{y}(k) \end{bmatrix}, \Phi \right), \tag{3.3}$$

and initialized using $\hat{x}(k = 0|\Phi) = 0$ and $\hat{y}(k = 0) = y_o$, where y_o represent the observable initial condition of truck dynamics.

Variable instantiations are detailed for each respective experiment in the later sections; however, we assume in general, for longitudinal dynamics,

$$u(k) = \begin{bmatrix} E_{\text{cmd}}(k) \\ B_{\text{cmd}}(k) \end{bmatrix}$$

$$y(k) = \begin{bmatrix} v(k) \\ a(k) \\ f_{\text{rate}}(k) \end{bmatrix}$$

$$w(k) = \theta_{\text{rdg}}(k),$$

where $E_{\text{cmd}}(k)$ is engine command, $B_{\text{cmd}}(k)$ is brake command, $v(t)$ is vehicle speed, $a(k)$ is vehicle acceleration, $f_{\text{rate}}(k)$ is fuel rate, and $\theta_{\text{rdg}}(k)$ is road grade each at discrete time step k.

3.3 Deep model

This section presents the f_{DT} model structure used to represent the longitudinal dynamics of the heavy duty truck. The model assumes that only controllable inputs to the truck, uncontrollable driving environment variables, and truck responses are known and measurable while the configuration of the truck and relevant internal state variables are not specified.

The state model,

$$x(k+1) = H(u(k), w(k)|x(k), y(k)),$$

represents an integrated state observer and tracker equations, and state update and encoder equations. The state model, $H(\cdot)$, is represented in this chapter as a long short-term memory (LSTM) recurrent neural network (RNN). The use of a single deep network unit to represent this model enables parameter sharing for the state observer, tracker, updater, and encoder functions.

The output model,

$$y(k) = G(x(k)),$$

represents an integrated state decoder equation and explicit output constraints model. The output model is represented by a cascade of a state decoder D and an explicit constraints models C such that $y(k) = C(D(x(k)))$. The state decoder is implemented as a fully connected feedforward neural network parameterized by Φ_{nn}. In the remainder of this chapter, we implement discrete-time longitudinal kinematics model as an explicit constraint model for longitudinal response output variables:

$$v(k+1) = v(k) + a(k) \cdot dt,$$

where $v(k)$ and $a(k)$ are longitudinal velocity and acceleration respectively, and dt is discrete time step. Other variables were left unconstrained.

3.4 Driving cycles for data collection

In this chapter, we assume that the internal dynamics of trucks can be observed from datasets were $y = (v, a)$, $u = (E_{\text{cmd}}, B_{\text{cmd}})$, and $w = \theta_{\text{rdg}}$ are jointly spanning. We consider that, without specialized driving data collection facilities, a human driver is most practical for data collection. Internal truck control signals u are often not accessible though the human driver interface (pedals), however, but are processed though vehicle manufacturer proprietary control systems as shown in Figure 3.4. We thus approximate such spanning dataset by a driving cycle consisting of (1) w and y spanning arbitrary acceleration/deceleration profiles, (2) w and v spanning coasting ($u = 0$), and (3) w and B_{cmd} spanning braking to zero speed.

For the field experiments presented in this chapter, these instructions were given to a human driver to execute for data collection. In simulation, we used a random generative model to approximate such a driving cycle. The generative model utilizes a moving average random walk model as a road profile generator. Coasting and braking to zero episodes are generated using direct randomized initializations. The spanning arbitrary acceleration/deceleration profiles were generated from the model presented in the next section.

Generative model for state space spanning driving cycles

For the numerical experiments presented in this chapter, we simulate arbitrary driving cycles using a speed profile generative model based on a time adaptive unstable stochastic speed controller. We designed it as a random speed profile (driving cycle) generator that samples the state space "fairly" uniformly.

Double integrator model is used with hard saturation limit at desired maximum and minimum speeds as follows:

$$v(t) = \max(\min(v(t-dt) + a(t) \cdot dt, v_{\max}), v_{\min}),$$

where $v_{\min}$ and $v_{\max}$ are desired minimum and maximum speeds of generated profile. Acceleration is sampled from a normal distribution as:

$$a(t) = \mathcal{N}(\mu_{\text{a, scaling}} \cdot \mu_{\text{a}}(t), \sigma_{\text{a, scaling}} \cdot \sigma_{\text{a}}(t)),$$

where $\mu_{\text{a, scaling}}$ and $\sigma_{\text{a, scaling}}$ are tuning parameters.

Acceleration statistics are designed based on speed dependant unstable feedback control. Average acceleration is given by:

$$\mu_{\text{a}}(t) = 1 - \frac{v(T_i)}{v_{\text{ref}}},$$

where v_{ref} is control reference speed, here set to $\frac{v_{\min}+v_{\max}}{2}$. The standard deviation is designed to allow for bursts of spontaneous high accelerations but discourage it at extreme speeds ($v_{\min}$, and $v_{\max}$) as follows:

$$\sigma_{\text{a}}(t) = \frac{v(T_i)}{v_{\text{ref}}} \cdot \left(1 - \frac{v(T_i)}{v_{\max}}\right).$$

T_i is used for acceleration-based adaptive temporal discretization for sampling acceleration statistics, and is designed to make high acceleration episodes short lived. The non-negative integer index is updated to $i := i + 1$ and T_{i+1} is re-sampled when t equals T_{i+1} and integrated as follows:

$$T_{i+1} = T_i + \lceil \max(\mathcal{N}(\mu_T) \cdot (1 - |\mu_{\text{a}}(t)|), dt) \rceil,$$

where μ_T is a tuning parameter.

To smooth out the noise, we pass the generated acceleration signal though a moving average filter to get $a_f(t)$, and re-integrate speed with a softmax operator as follows:

$$v_f(t) = \frac{\max(v_f(t-1) + a_f(t) \cdot dt, v_{\min})}{1 + e^{\frac{1}{2} \cdot (\max(v_f(t-1)+a_f(t) \cdot dt, v_{\min}) - v_{\max})}},$$

noting that acceleration signal has to be recalculated, as needed, from speed signal after this step.

3.5 Deep-RL continuous longitudinal control

In this chapter, we use deep-reinforcement-learning to design end-to-end heavy duty truck controllers allowing for offline (1) design/tune the controller, (2) calibrate the controller module to the specific physics of each truck, (3) design an embedded state observer/tracker. In developing these controls we assume limited observable input and output (IO), unknown truck mechanical configuration, and unknown relevant internal state. We formulate the problem as a *Partially Observable Markov Decision Processes* (POMDP) and solve it using deep reinforcement learning framework as discussed in Chapter 2.

Deep-RL cooperative adaptive cruise control

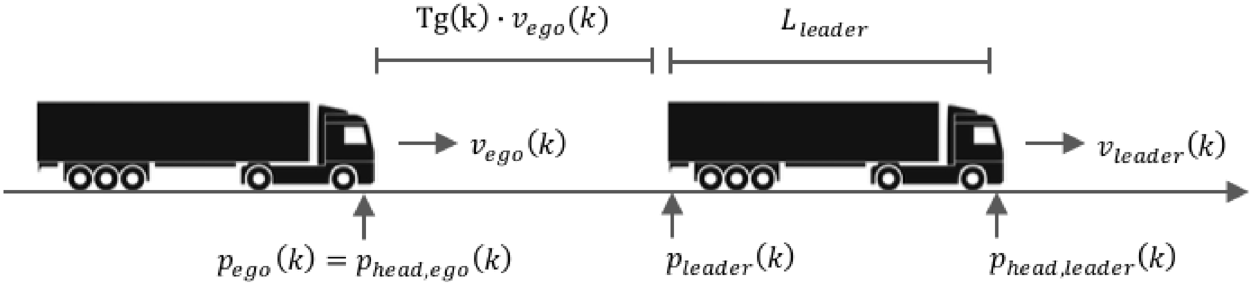


Figure 3.2: Two truck cooperative adaptive cruise control system setup.

This section formulates an end-to-end two-truck cooperative adaptive cruise control (CACC) [15] using deep-RL based on the longitudinal truck model developed in this chapter. In this system, we consider a human driven leader vehicle, and a follower semi-automated to simultaneously regulate speed and time gap as shown in Figure 3.2.

The environment is modeled by a two point mass system representing each of the two trucks. Both vehicle dynamics were modeled using the double integrator kinematic model. Leader vehicle (leader) dynamics were simplified as a linear system. The velocity of the controlled truck (ego) is modeled as a nonlinear system according to the deep truck model

f_{DT} presented earlier in this chapter. This results in the following environment model:

$$\begin{aligned}
\begin{bmatrix} p_{\text{leader}}(k+1) \\ v_{\text{leader}}(k+1) \\ p_{\text{ego}}(k+1) \\ v_{\text{ego}}(k+1) \end{bmatrix} &= \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{\text{leader}}(k) \\ v_{\text{leader}}(k) \\ p_{\text{ego}}(k) \\ v_{\text{ego}}(k) \end{bmatrix} \\
&+ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \cdot a_{\text{leader}}(k) \cdot dt \\
&+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot f_{DT,ego,v} \left(\begin{bmatrix} E_{\text{cmd, ego}}(k) \\ B_{\text{cmd, ego}}(k) \\ \theta_{\text{rdg, ego}}(k) \end{bmatrix} \middle\| \begin{bmatrix} x_{\text{ego}}(k|\Phi) \\ v_{\text{ego}}(k) \end{bmatrix}, \Phi \right),
\end{aligned}$$

where $f_{DT,\text{ego},v}$ represents the velocity component from the deep truck model for the ego truck, and relevant variables ($E_{\text{cmd, ego}}$, $B_{\text{cmd, ego}}$, $\theta_{\text{rdg, ego}}$, x_{ego}, and Φ) are as defined in Section 3.2. As shown in Figure 3.2, p_{leader}, v_{leader}, and a_{leader} are absolute longitudinal position, velocity, and acceleration representing the leading vehicle, and p_{ego} and v_{ego} are absolute longitudinal position and velocity of the ego vehicle. Time step size is represented by dt.

The agent is represented by the probability distribution function $\pi(a_k|o_k, \Phi_{\text{agent}})$, where a_k represent agent action, o_k represent observation at time step k, and Φ_{agent} represent agent parameters. The corresponding control $u(k)$ is implemented as:

$$\begin{aligned}
u(k) = \begin{bmatrix} E_{\text{cmd, ego}}(k) \\ B_{\text{cmd, ego}}(k) \end{bmatrix} &= f_{\pi} \left(\begin{bmatrix} v_{\text{leader}}(k) \\ v_{\text{ego}}(k) \\ p_{\text{leader}}(k) - p_{\text{ego}}(k) \\ v_{\text{ego}}(k) \cdot Tg_{\text{target}} \\ \theta_{\text{rdg}}(k) \end{bmatrix} \right) \\
&= E \left(\pi \left(a_k \middle| o_k = \begin{bmatrix} v_{\text{leader}}(k) \\ v_{\text{ego}}(k) \\ p_{\text{leader}}(k) - p_{\text{ego}}(k) \\ v_{\text{ego}}(k) \cdot Tg_{\text{target}} \\ \theta_{\text{rdg}}(k) \end{bmatrix}, \Phi_{\text{agent}} \right) \right)
\end{aligned}$$

representing the mean value for a Multi-Layer Perceptron (MLP) Gaussian distribution model.

The reward function is designed to simultaneously regulate time-gap between ego and leader to a given desired time-gap, and regulate velocity of ego to match that the leader. The agent is penalized for actuation cost, here approximated by engine and brake commands. Safety constraint is implemented as a very large penalty term applied when minimum safety distance is violated. The reward function is modeled as:

$$\begin{aligned} r(k) = -\alpha_p(p_{\text{leader}}(k) - p_{\text{ego}}(k) - v_{\text{ego}}(k) \cdot Tg_{\text{target}})^2 \\ - \alpha_v(v_{\text{leader}}(k) - v_{\text{ego}}(k))^2 - \alpha_E E_{\text{cmd}}^2(k) - \alpha_B B_{\text{cmd}}^2(k) \\ - \alpha_{\text{crash}} \cdot (p_{\text{leader}}(k) - p_{\text{ego}}(k) \leq d_{\text{safety}}), \end{aligned}$$

where $Tg(k)$ is actual time-gap between leader tail and ego head, Tg_{target} is target (desired) time-gap, α_x, α_v, α_E, and α_B are positive constants, α_{crash} is a large positive constant, d_{safety} is minimum safety distance, and all other variables are as defined earlier in this section.

Each training episode is initialized using leader position $p_{\text{leader}}(k = 0) = 0$, random initial ego truck position error $p_{\text{leader}}(k = 0) - p_{\text{ego}}(k = 0) - v_{\text{ego}}(k = 0) \cdot Tg_{\text{target}}$ from a uniform$(p_{o,min}, p_{o,max})$, random initial leader speed $v_{\text{leader}}(k = 0)$ from uniform$(v_{o,min}, v_{o,max})$, random initial ego truck speed error $v_{\text{ego}}(k = 0) - v_{\text{leader}}(k = 0)$ from uniform$(v_{o,min}, v_{o,max})$ distribution, random desired time gap Tg_{target} from a uniform$(Tg_{o,min}, Tg_{o,max})$ distribution, and random constant road grade θ_{rdg} from a uniform$(\theta_{\text{rdg, o, min}}, \theta_{\text{rdg, o, max}})$ distribution. To simplify the setup, we also assume $a_{\text{leader}}(k) = 0$. All distribution boundaries are positive constants chosen to cover the desired operational state-space of the CACC system and be constrained by the state-space covered by the deep model where appropriate.

3.6 Vehicle pool

We primarily utilize three trucks with three different mechanical configurations for the study presented in this chapter as shown in shown in Figure 3.1 and Figure 3.3. One truck is simulation based used primarily for numerical experiments. The remaining two trucks are full-size real-physical trucks that had been modeled using two different physics-based powertrain models in [14] and [15] and used to develop high precision control systems within each respective article.

Simulation framework and simulation truck mechanical configuration. Simulation experiments in this chapter are conducted in TruckSim [74], a black-box state-of-the-art commercial software framework with high fidelity modeling capabilities and a detailed vehicle and vehicle component libraries.

The truck, shown in Figure 3.3, is equipped with a 402hp engine. The engine shaft is connected to one side of the transmission via clutch. The clutch allows speed difference between the engine and the transmission when gear shifts. The transmission has ten forward gears and one reverse gear. The other side of the transmission is connected to rear wheels via a differential gear with a fixed reduction ratio. The truck is equipped with an air-brake system. The front air-brakes have capacity of 7.5 kN-m on each wheel. The rear brakes have capacity of 10 kN-m on each wheel. Actuation control input to the truck are engine torque and brake cylinder pressure. The details are presented here for completeness and for reporting purposes, but are irrelevant to the deep model.

Real full-size Freightliner truck mechanical configuration. The Freightliner truck used for the results in this section is a tractor-only Freightliner Century truck driven

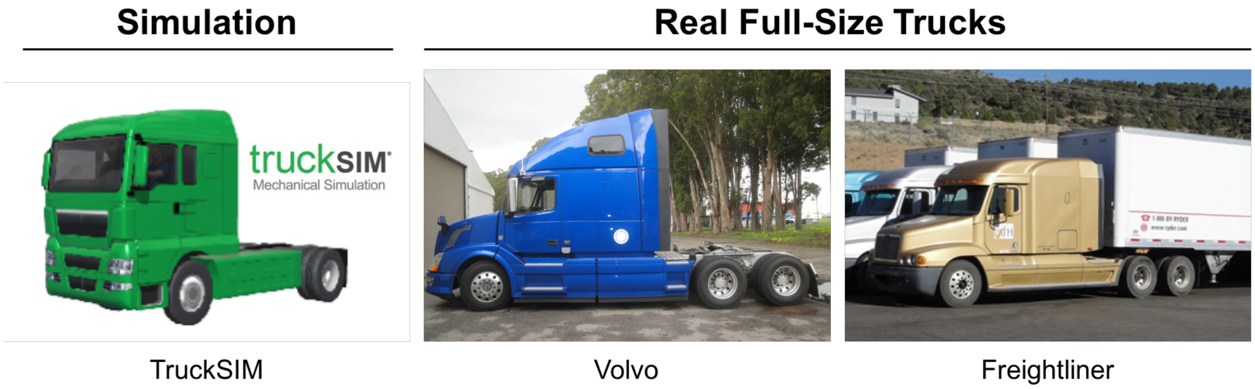


Figure 3.3: Real full-size and simulations trucks of multiple mechanical configurations used in this research.

by a 435 hp turbocharged Detroit Diesel diesel engine and equipped with a 6 gear true-automatic (equipped with torque-converter) Allison transmission system. The service brake is a drive by wire all the way to the wheels. The truck is not equipped with road grade sensors.

Real full-size Volvo truck mechanical configuration. The second set of experiments were conducted using a Volvo VNL truck (with and without a tractor) driven by a 500 hp engine. The mechanically most significant differentiator of this truck from the Freightliner truck is the transmission system which is an automated manual-transmission (equipped with clutches).

3.7 Vehicle interface

Access to vehicle powertrain is often primarily provided through a human driver interface (pedals) and is mediated by proprietary controllers as shown in Figure 3.4. For precision sensitive applications however, it is often desirable to probe as close to the powertrain as possible (e.g. engine torque or engine fuel rate control signals). We access these signals through a custom-built automated driver interface connected to vehicle communication backbone J-1939. The interface provide access to powertrain and sensor signals; however, architectural details and signal accessibility vary between truck platforms. Multiple layers of fail-safe safety systems were implemented to ensure experiments remain faithful to published description while maintaining safety on the road. Parallel interfaces and system architecture is used for the simulation truck.

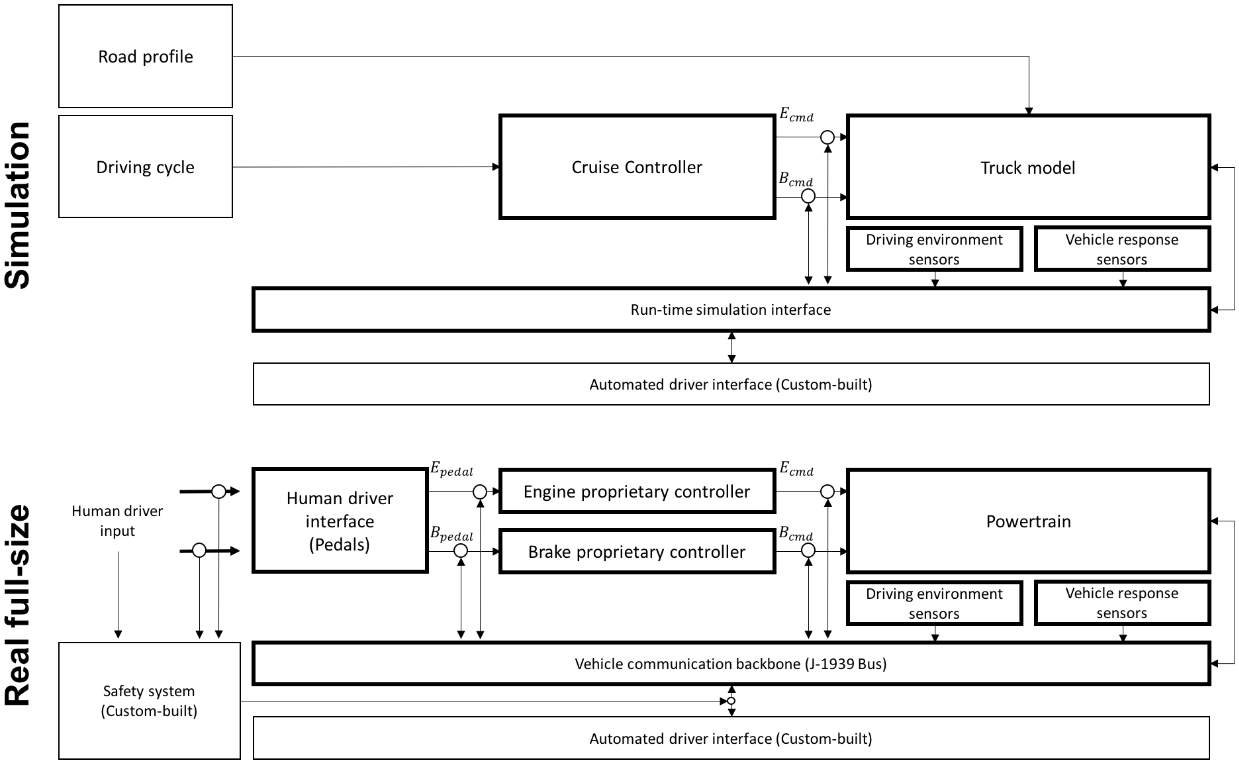

Simulation
Road profile
Driving cycle
Cruise Controller
E_{cmd}
B_{cmd}
Truck model
Driving environment sensors
Vehicle response sensors
Run-time simulation interface
Automated driver interface (Custom-built)
Real full-size
Human driver input
Human driver interface (Pedals)
E_{pedal}
B_{pedal}
Engine proprietary controller
Brake proprietary controller
E_{cmd}
B_{cmd}
Powertrain
Driving environment sensors
Vehicle response sensors
Vehicle communication backbone (J-1939 Bus)
Safety system (Custom-built)
Automated driver interface (Custom-built)

Figure 3.4: Interface architecture for deep modeling and control of heavy duty trucks.

3.8 Experiments

This section presents experimental evaluation of the process detailed in this chapter. The section starts by applying the process to a simulation based truck to present detailed performance statistics. The section then reapplies the process to full-size trucks.

Deep modeling of the simulation truck

This section presents experimental results for the development of a deep learning model as described in Section 3.2 for the simulation truck.

Deep model specifications

In this experiment, the uncontrollable conditions $w(k) = \theta_{\mathrm{rdg}}(k)[\%]$ represent road grade. The controllable input to the truck is given by $u(k) = [E_{\mathrm{cmd}}(k), B_{\mathrm{cmd}}(k)]$, where $E_{\mathrm{cmd}}(k)$ is engine torque in $[N-m]$ and $B_{\mathrm{cmd}}(k)$ is service brake master cylinder pressure $[0-100\%]$.

The output (truck response) vector is given by $y(k) = [a(k), v(k), F_{\mathrm{rate}}(k)]$, where $a(k)$ is longitudinal acceleration in $[m/s^2]$, $v(k)$ is longitudinal speed in $[m/s]$, and $F_{\mathrm{rate}}(k)$ is fuel rate in $[cm^3/s]$.

Driving datasets

For training, we simulated a total of four hours of driving using the data collection strategy presented in Section 3.4. We generated another three hour set for testing and validation to evaluate modeling performance on unseen data. All datasets span speeds from zero to 35 m/s and road grades from ±3%. A sample of the dataset is shown in Figure 3.5 and Figure 3.6.

Model learning curves

This section presents learning curves for training the deep truck model for the simulation truck using the data presented in this section. Figure 3.7 shows the loss function statistics (mean, min and max) based on training form Equation (3.2) and deployment form Equation (3.3) from 30 seeds. Each curve is produced using a separate dataset both unseen during training.

Both learning curves stabilize and converge by the 600th epoch. They exhibits spikes we speculate are a symptom of the inherent stochasticity of the mini-batch algorithm we used. An expected loss gap between training form curve and deployment form curve is observed.

Results and model validation

This section presents model validation results using an unseen validation dataset. Figure 3.8 shows modeling error statistics as a function of model simulation time from 90 independent random trails. Error statistics are generated as:

$$\text{ErrorStatistic}(k) = \text{Statistic}_m \left(\hat{y}_m(k|\Phi) - y_m(k)\right)$$

where m is trial number, and $\hat{y}$ follow the deployment form Equation (3.3). Distributions (initial speed distribution, visited speed over time and across all trials, visited road grades over time and across all trials) of the validation dataset are shown in Figure 3.9.

Mean of modeling error stays bounded near zero over the 40 second simulation time. On average acceleration deviates by less than $0.5m/s^2$ and fuel deviates by less than 10^{-3} at any given time. The statistics also show that the error of modeled speed is expected to remain within $1.5m/s$ over a 40 second simulation time.

A sample model validation dataset is shown in Figure 3.10 and Figure 3.11. In this validation experiment, the model is initialized once at $k = 0$ and then simulated for 2000 time steps ($t_{\text{end}} = 200s$). The dataset exhibits a large initial error transient with significant model response delay estimation error. Error statistics appear to be (by visual inspection) stationary consistent with error statistics in Figure 3.8.

Deep-RL control of the simulation truck

This section presents experimental results for the development of a deep-RL CACC as described in Section 3.5 for the simulation truck.

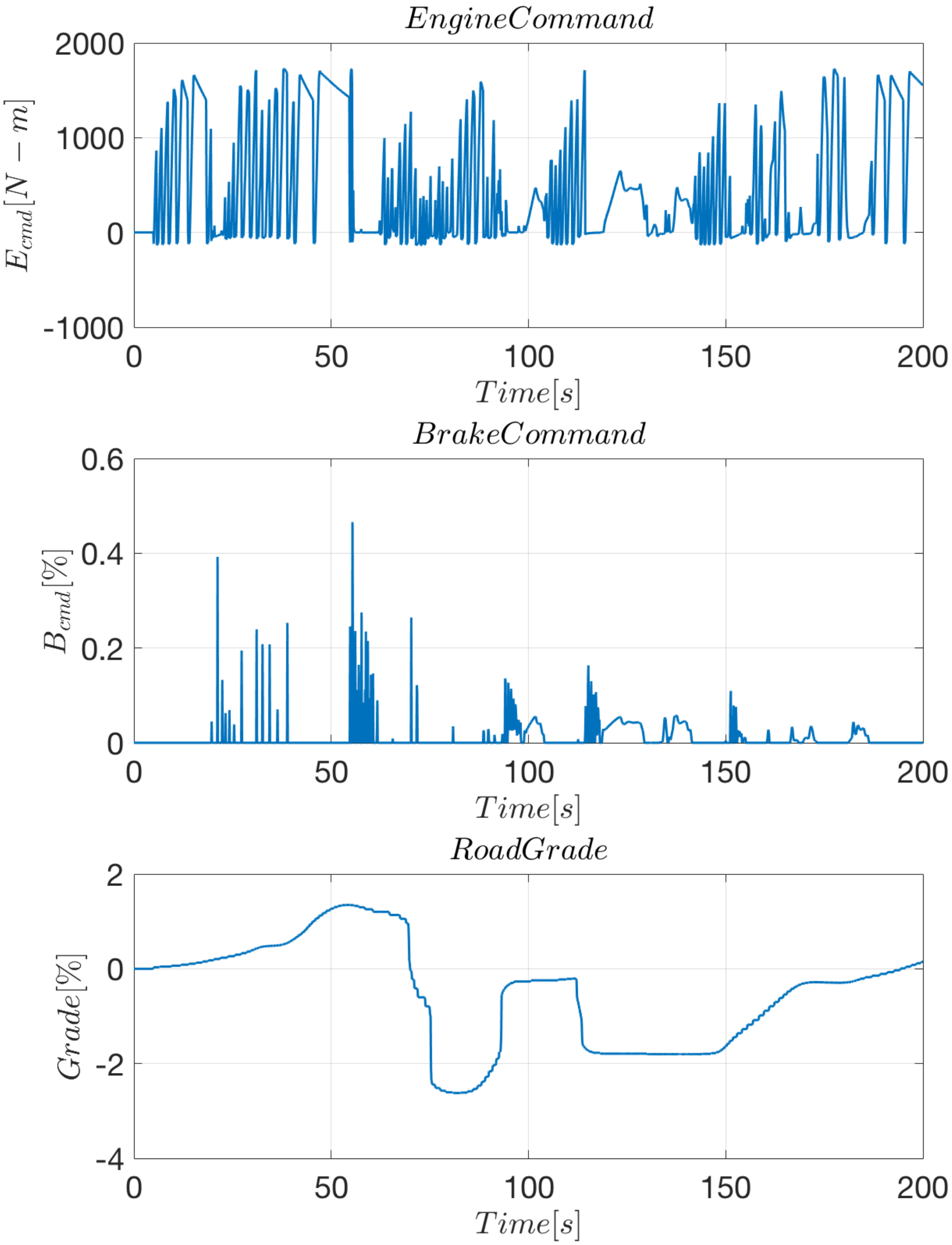


Figure 3.5: A ground truth sample dataset representing inputs to the deep model.

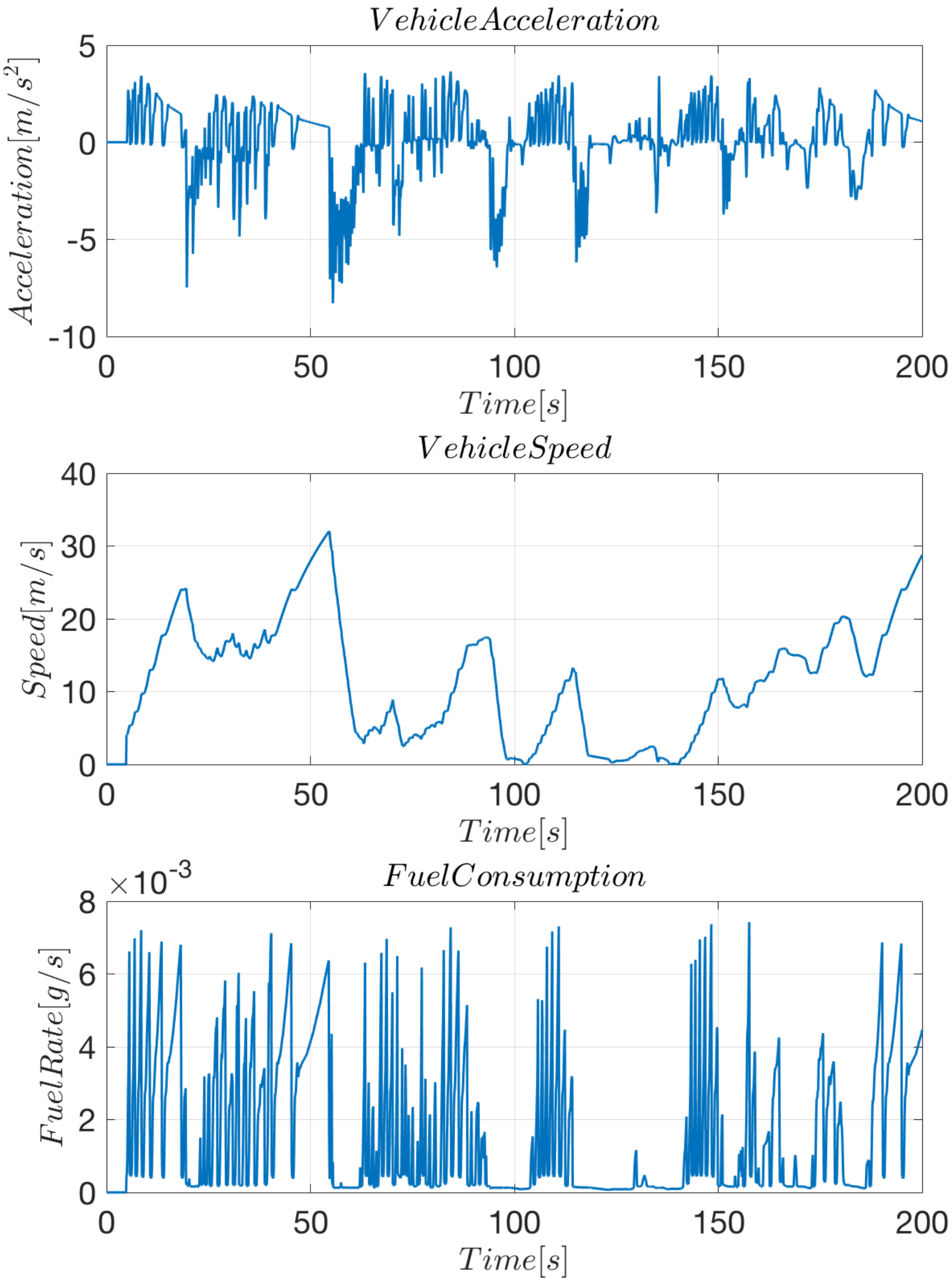


Figure 3.6: A ground truth sample dataset representing outputs from the deep model.

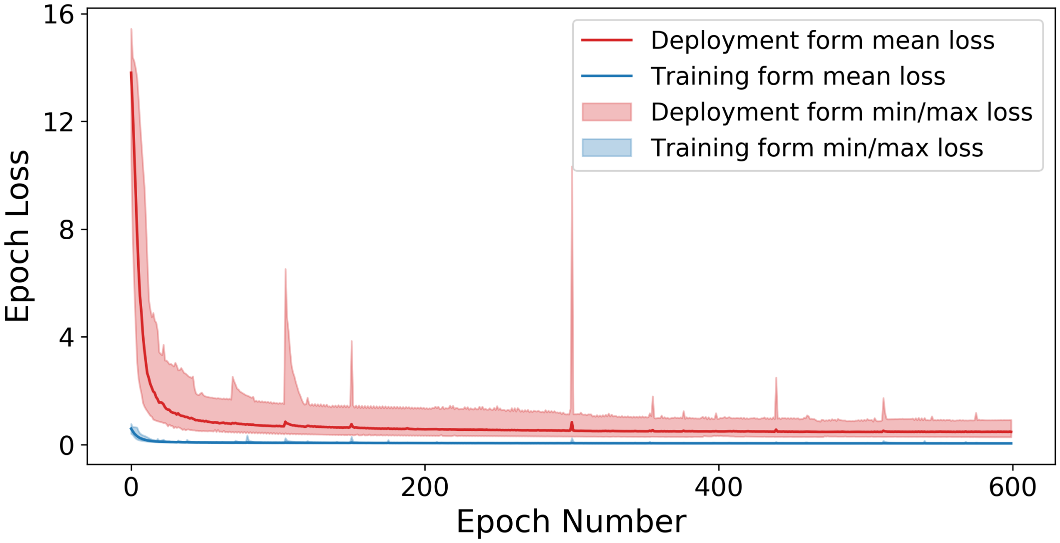


Figure 3.7: Learning curve—min/max and mean from 30 seeds—for deep modeling of the TruckSIM truck.

Training setup and learning curves

For this experiment, the sampling rate is set to $10Hz$ ($dt = 0.1s$) and we assume flat driving environment with no other relevant driving environment variables; thus we substitute $w(k)$ with the empty set. The controllable input to the ego truck (agent output) is given by $u(k) = [E_{\text{cmd, ego}}(k), B_{\text{cmd, ego}}(k)]$, where $E_{\text{cmd, ego}}(k)$ is requested engine torque in $[N - m]$ and $B_{\text{cmd, ego}}(k)$ is requested service brake master cylinder pressure percentage $[0 - 100\%]$. The agent π is modeled using an ANN that has 3 hidden layers, each of size 25.

Each training episode is initialized using $p_{\text{leader}}(k = 0) = 0$, random initial ego truck position $p_{\text{ego}}(k = 0)$ from a $-(v_{\text{ego}}(k = 0) \cdot Tg_{\text{target}} + \text{uniform}(-1.39, 1.39))$, random initial leader speed $v_{\text{leader}}(k = 0)$ from $\text{uniform}(8.3, 22.2)$ and $v_{\text{ego}}(k = 0)$ from $v_{\text{leader}}(k = 0) + \text{uniform}(-1.39, 1.39)$ distributions, and random desired time gap Tg from a $\text{uniform}(2, 5)$ distribution. To simplify the setup, we also assume $a_{\text{leader}}(k) = 0$.

The deep-RL controller was trained on RLLab [84] using batch size of 20000, max path length of 800 (sampled at 10Hz) and discount factor of 0.9999. We trained ten policies (ten seeds). The average discounted returns plot is shown in Figure 3.12. The trained policies shown in the plot converged after 500 iteration. The observed sharp numerical negative infinity return values are caused by crashes between the two trucks inside the training environment as specified by the reward function presented in Section 3.5. These crashes happen as the agent of the deep reinforcement learning explores the state-action space, which is implemented here by means of a stochastic agent policy.

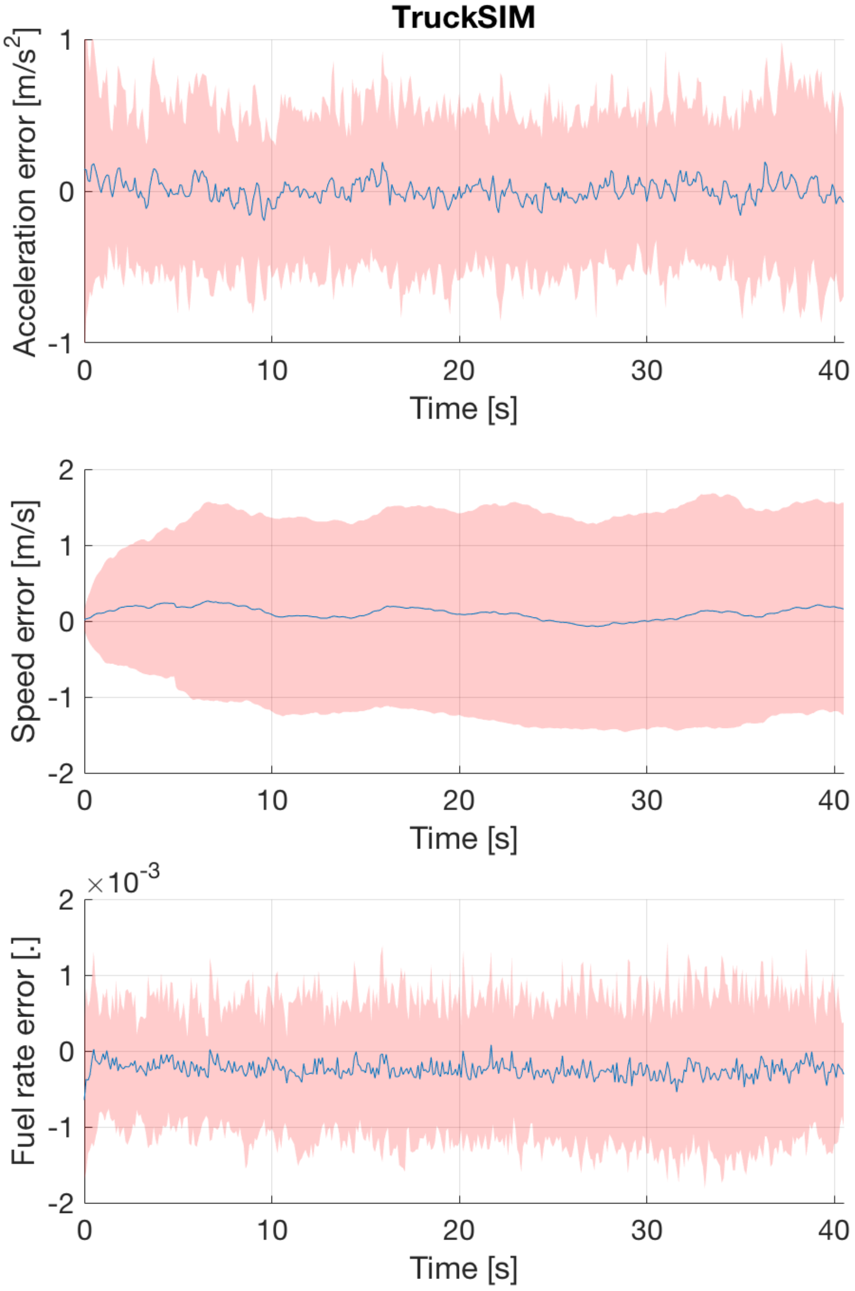


Figure 3.8: Model error statistics—standard deviation (red shaded areas) and mean (blue curves) from 90 trials—for deep modeling of the TruckSIM truck.

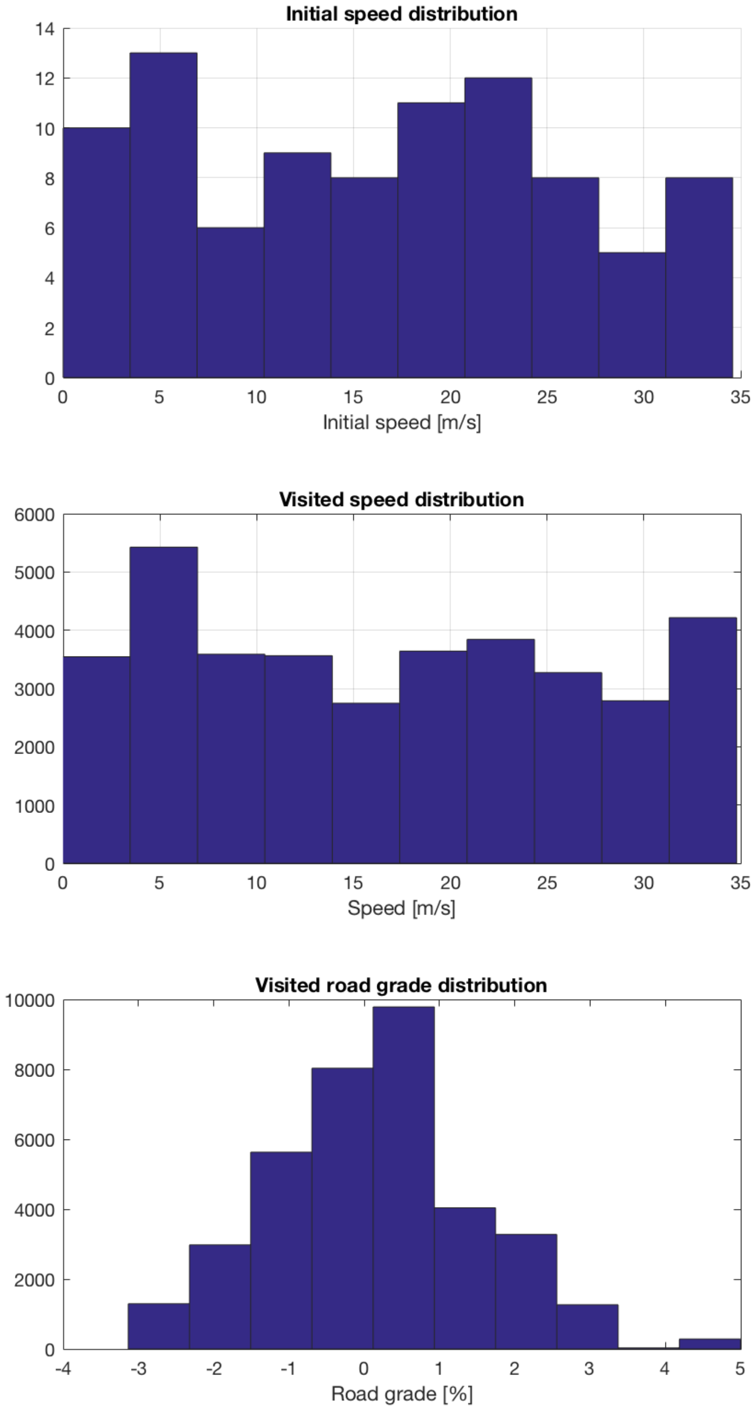


Figure 3.9: Scenario distribution of the dataset used to compute model validation statistics presented in Figure 3.8.

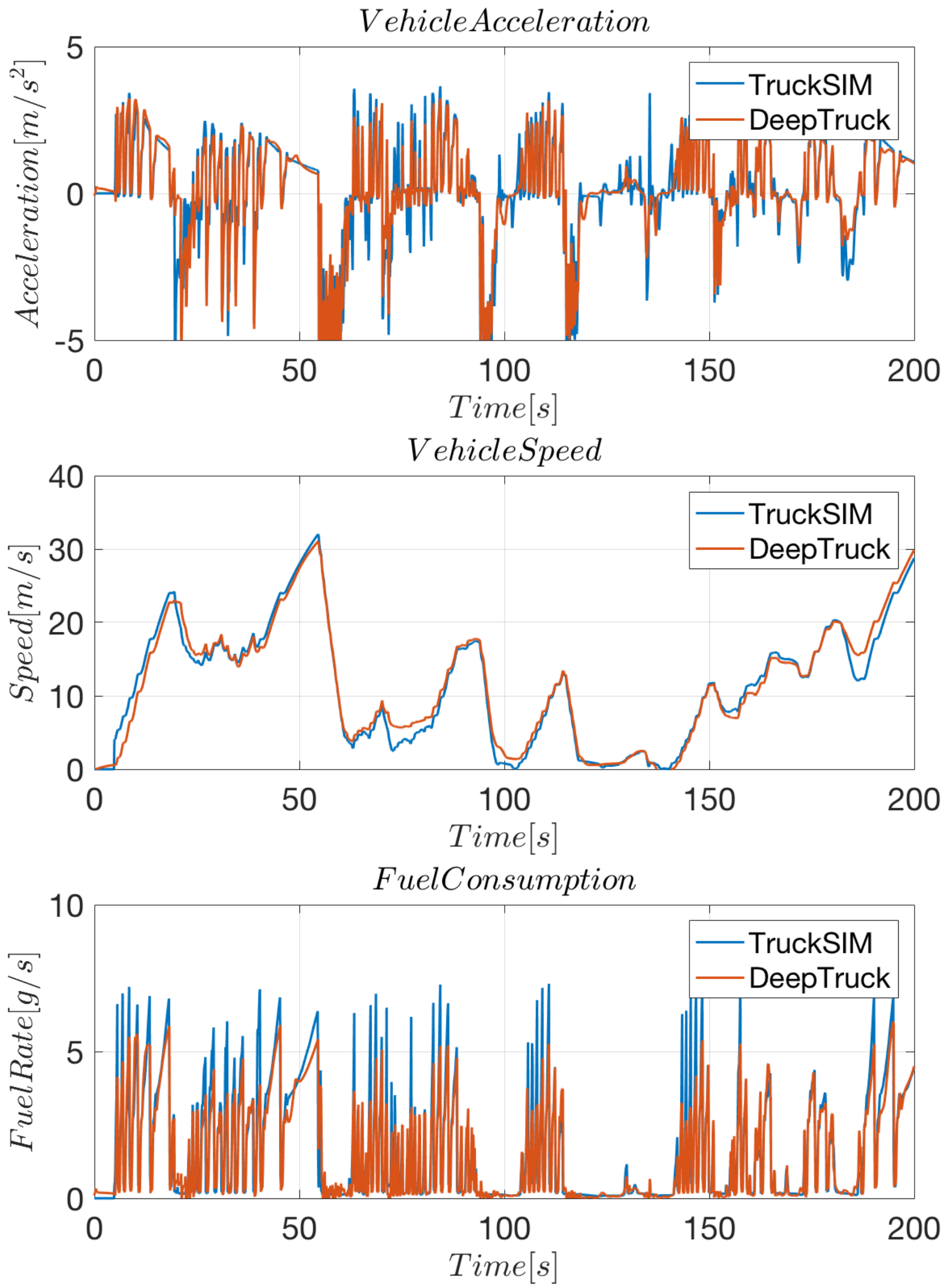


Figure 3.10: A sample ground truth and model output using a sample validation set.

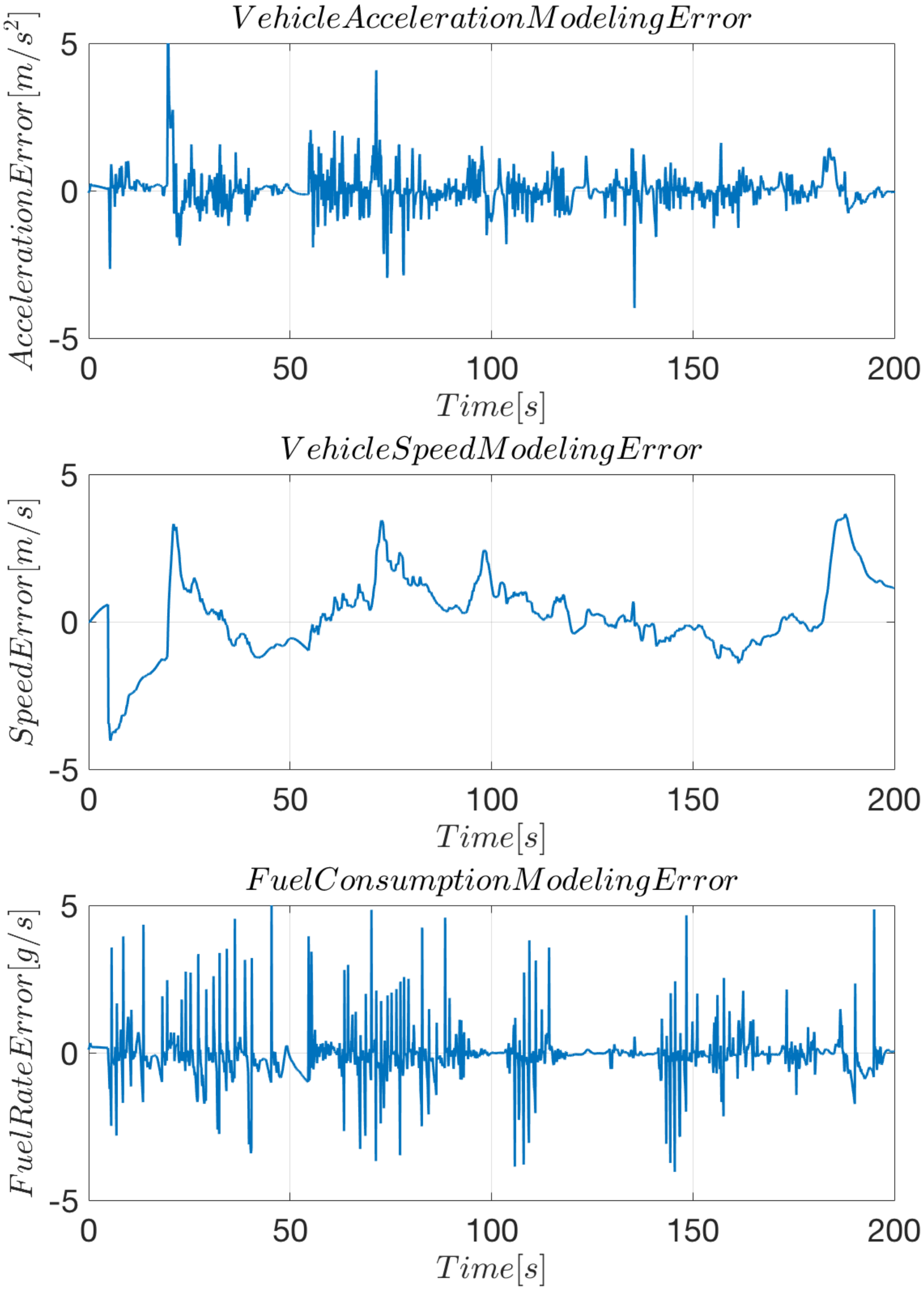


Figure 3.11: A sample error over time between ground truth and model output using a sample validation set.

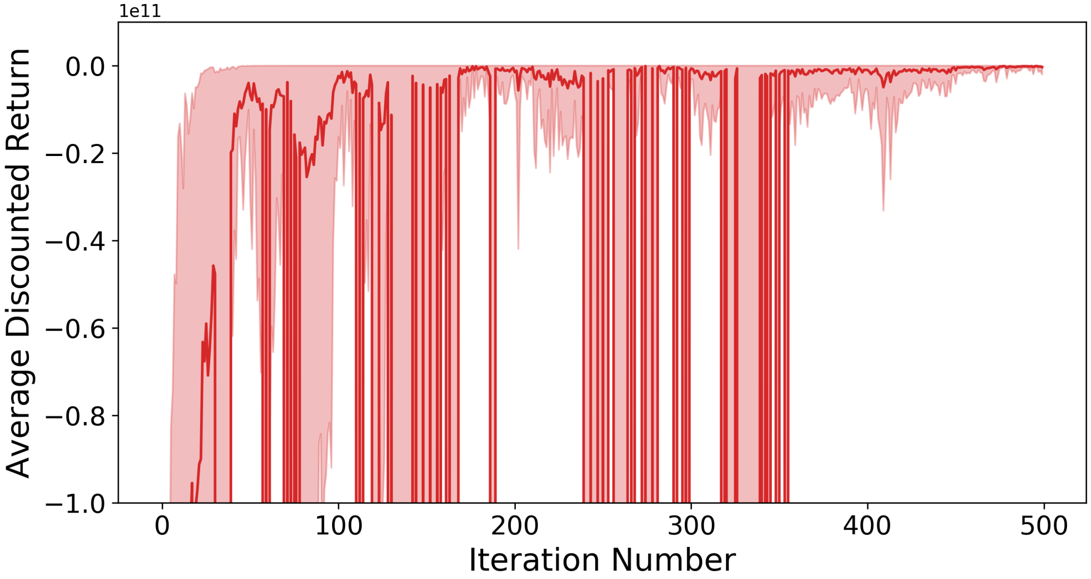


Figure 3.12: Learning curve—min/max (light red shaded area) and mean (dark red curve) from ten seeds—for deep cruise control policy based on deep model of the TruckSIM truck.

Control validation results

This section validates the performance of the deep-RL cooperative adaptive cruise controller against both the deep environment and transfer into TruckSIM as shwon in Figure 3.13. DeepEnv-set experiment is a replication of the training setup and consists of 100 rollouts drawn from the same training distributions (environment model and initialization distributions). The same controller is zero-shot transferred to TruckSIM to produce TruckSIM-set consisting of 10 rollouts drawn from the same initialization distributions.

The policy is designed to simultaneously regulate speed and time-gap. In DeepEnv-set, time-gap error converges to steady state error between $\pm 0.05s$ within $10s$ from the start time of the experiment, while speed error converges to steady state error between $\pm 0.03m/s$ within $25s$ from the start time of the experiment both with mean error of approximately zero.

TruckSIM-set evaluates the transfer of the same policy to TruckSIM environment with the same random distributions. Observed shift in control performance is caused by shift in truck model distribution due to modal mismatch discussed in the modeling experiments. Time-gap error converges to steady state error between $0.04s$ and $-0.2s$ within $10s$ from the start time of the experiment, while speed error converges to steady state error between $\pm 0.02m/s$ within $25s$ from the start time. The time-gap mean error converges to $-0.06s$, while speed mean error converges to approximately zero. The controller exhibits a nonlinear bimodal speed control over/undershoot.

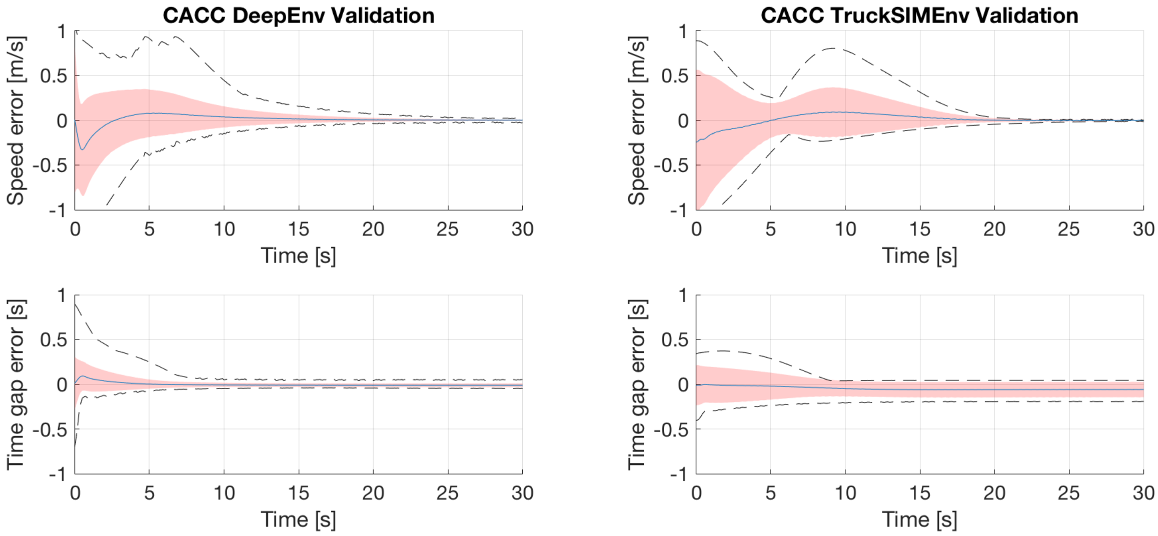


Figure 3.13: Control error statistics—min/max (dashed curves), standard deviation (red shaded areas) and mean (blue curves)—for the deep policy evaluated against the deep environment and against TruckSIM environment.

Figure 3.13 shows preliminary learning results for deep-RL cooperative adaptive cruise controller and shows preliminary evidence to expect marginal shifts in error statistics when transferring the policy from the deep-truck environment to the "real" environment (here conducted using a simulated truck).

Deep modeling of full-size trucks (field experiments)

This section presents field experimental results for the model described in this chapter. The section documents experiments conducted using two differently configured real-physical heavy duty trucks. These same two trucks were modeled using two different physics-based power-train models in [14] and [15] used to develop high precision control systems within each respective article.

Configuration one: Freightliner

In this experiment, the truck is not equipped with any sensors relevant to the driving environment (e.g. road grade) and thus we substitute $w(k)$ with the empty set. The controllable input to the truck is given by $u(k) = [E_{\text{cmd}}(k), B_{\text{cmd}}(k)]$, where $E_{\text{cmd}}(k)$ is requested percentage engine torque in $[0 - 100\%]$ and $B_{\text{cmd}}(k)$ is service brake pedal position $[0 - 100\%]$.

The output (truck response) vector is given by $y(k) = [a(k), v(k), F_{\text{rate}}(k)]$, where $a(k)$ is longitudinal acceleration in $[m/s^2]$, $v(k)$ is longitudinal speed in $[m/s]$, and $F_{\text{rate}}(k)$ is fuel rate in $[cm^3/s]$.

Experiments for this truck has been carried out at a nearly flat test track with straight roads the longest of which is around 300 meters long at the Richmond Field Station at California. The truck was driven for about 16 minutes to collect primarily slow speed dataset covering from zero to $18m/s$. The dataset was split into 85 percent for training and 15 percent to test modeling performance on an unseen dataset.

Configuration two: Volvo

The truck is equipped with a road grade sensor where $w(k) = \theta_{\text{rdg}}(k)[\%]$. The controllable input to the truck is given by $u(k) = [E_{\text{cmd}}(k), B_{\text{cmd}}(k)]$, where $E_{\text{cmd}}(k)$ is requested percentage engine torque in $[0 - 100\%]$ and $B_{\text{cmd}}(k)$ is service brake command $[m/s^2]$.

Actuation and accessible signal measurement of the brake system in this truck is asymmetric. The service brake system in this truck is not directly actuatable (and signals not interceptable); instead, desired deceleration is processed through Volvo propriety systems. After collecting the data, we substitute brake commands with observed deceleration gated by brake pedal gating switch signal.

The output (truck response) vector is given by $y(k) = [a(k), v(k), F_{\text{rate}}(k)]$, where $a(k)$ is longitudinal acceleration in $[m/s^2]$, $v(k)$ is longitudinal speed in $[m/s]$, and $F_{\text{rate}}(k)$ is fuel rate in $[cm^3/s]$.

This truck was primarily driven over non-flat open freeways. The truck was driven for about 24 minutes to collect primarily freeway speed driving dataset covering speeds from $20m/s$ to $30m/s$. The dataset was split into 85 percent for training and 15 percent to test modeling performance on an unseen dataset.

Results and model validation

We validate modeling performance against an unseen ground truth dataset from each truck configuration as shown in Figure 3.14. In this figure, mean and standard deviation for acceleration, speed, and fuel rate modeling errors are charted as a function of model simulation time. The statistics were produced from an ensemble of ten timeseries simulations. Each simulation is fresh initialized at time zero, and simulated using knowledge of inputs and the uncontrollable conditions only. Error statistics are generated as $\text{ErrorStatistic}(k) = \text{Statistic}_m\left(\hat{y}_m(k|\Phi) - y_m(k)\right)$ where m is trial number, and $\hat{y}$ follow the deployment form Equation (3.3).

In this figure, acceleration error is bounded between $\pm 0.5m/s^2$. For the Freightliner, speed error remained bounded between $\pm 0.5m/s$ mean of speed error $0.12m/s$ after the initial transient. For the Volvo, speed error remained between $-0.5m/s$ and $1m/s$ with a significant error bias approaching $0.5m/s$ during the 15 seconds of simulation. Fuel rate modeling error is bounded between ± 1 once the initial transient decays. We speculate that model performance degradation for the Volvo truck is influenced by insufficient data to model truck dynamics over graded roads.

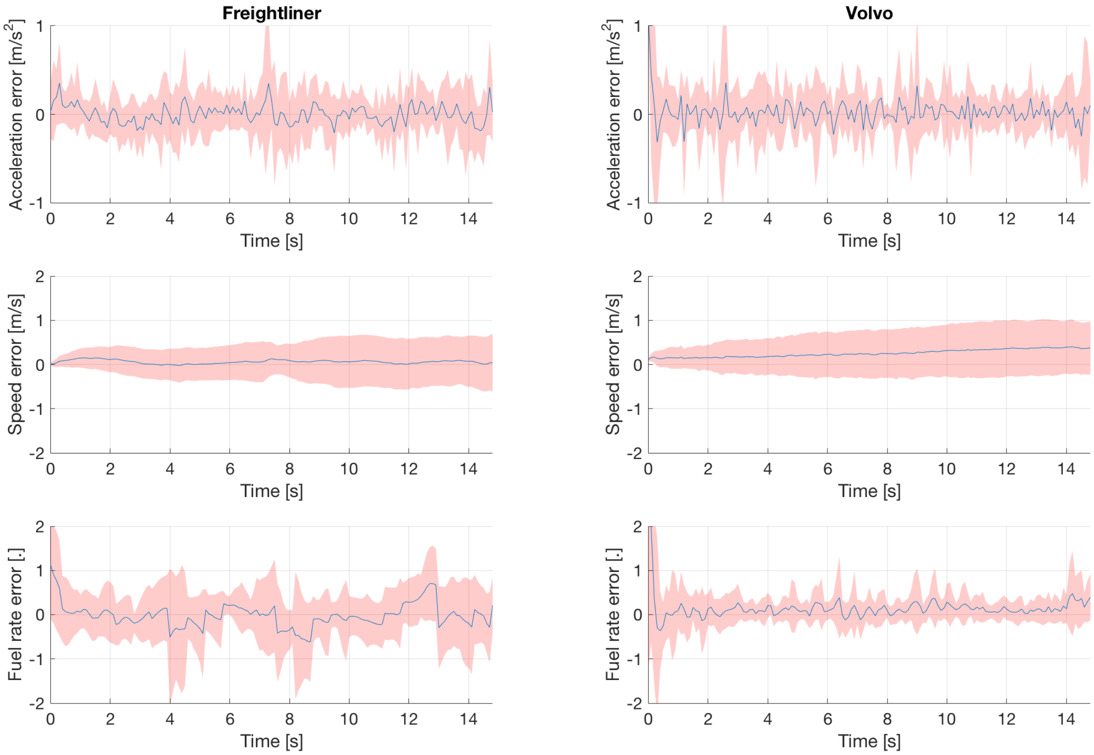


Figure 3.14: Model error statistics—standard deviation (red shaded areas) and mean (blue curves) from 10 trials—for deep modeling of the Freightliner and the Volvo trucks.

Deep-RL control of full-size trucks (field experiments)

This section presents control experiments for the deep-RL CACC system presented earlier. The system was operated as a two vehicle ACC (using radar instead of direct communications) system on non-flat open freeways. The leader is a passenger car and the follower is the Volvo truck presented earlier.

Figure 3.15 shows gap closing regulation performance where the leader drove at nearly constant speed with initial speed error of $2m/s$, initial time gap error of $1s$, and a desired time-gap setting of $1.5s$. The gap was closed within 15 seconds and to within error bound of $\pm 0.2m/s$ and $0.35s$. Leader conducted a quick successive changes of speed towards the end of experiment causing the observed speed ripple after time $17s$.

Figure 3.16 shows tracking performance over an arbitrary driving cycle conducted by the leader vehicle with a desired time-gap setting of $1.5s$. Speed error was regulated to within $\pm 0.5m/s$ and time gap was regulated to between $0.05m/s$ and $0.3m/s$. A lane change maneuver was conducted at time $63s$ causing a momentary misalignment between ego vehicle's sensor line-of-sight with the leader. Speed and distance measurements of a farther vehicle down stream was detected causing the observed discontinuity.

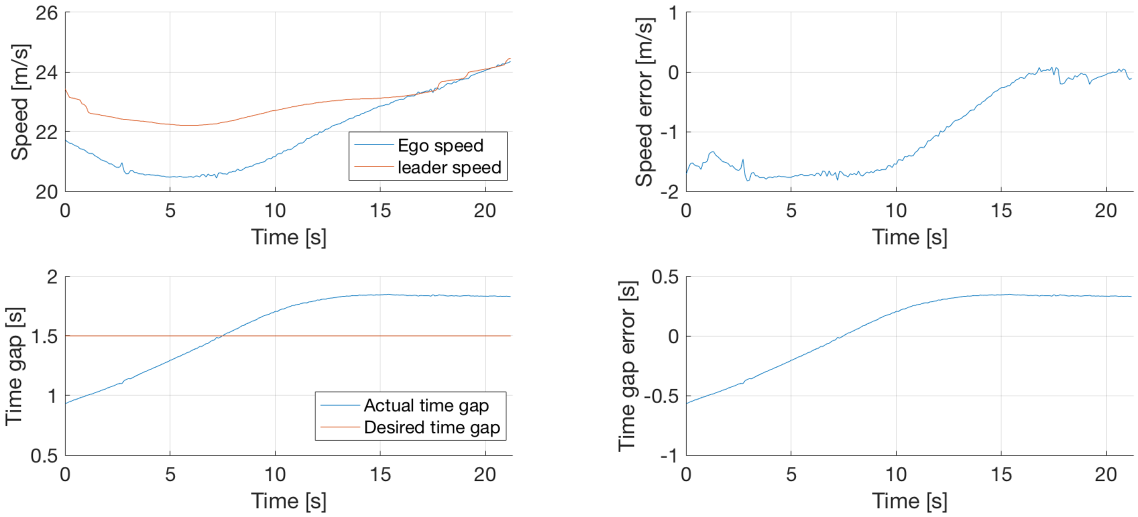


Figure 3.15: Tracking speed, time-gap, and control error for the Volvo deep CACC policy evaluated against the real environment—gap closing maneuver.

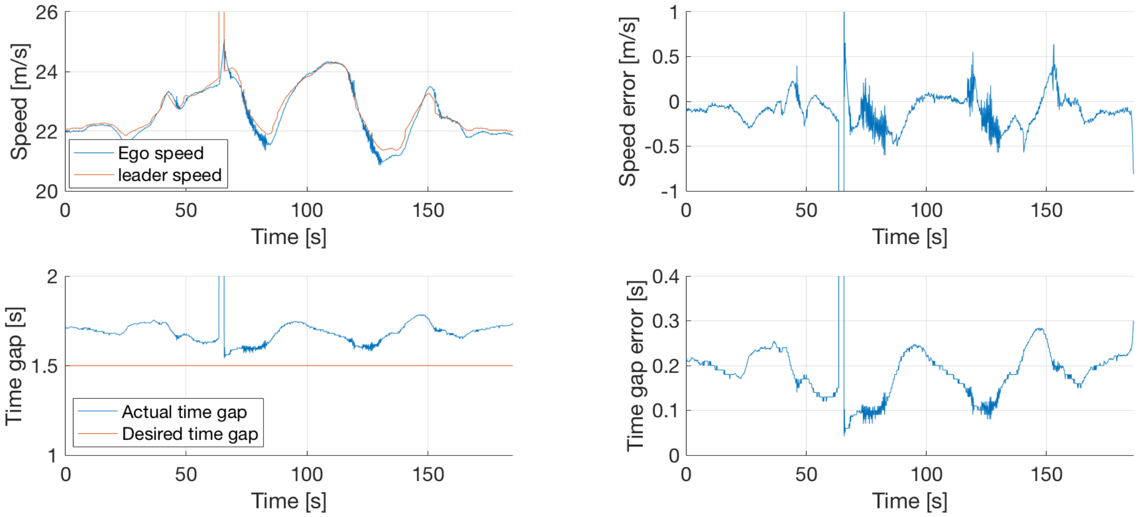


Figure 3.16: Tracking speed, time-gap, and control error for the Volvo deep CACC policy evaluated against the real environment—leader following maneuver.

3.9 Conclusion

Detailed study of each heavy duty truck in some pool of trucks has historically been required to develop and fit precise analytical models and controls. This chapter discusses the application of deep learning and deep reinforcement learning as an approach to simplify the process and abstract detailed vehicle underlying mechanics with a potential for improving modeling and control precision. A brief experimental evaluation is presented as a walk through the process and as preliminary performance validation.

The deep models and deep-RL controls presented in this chapter successfully (1) infers relevant latent and state variables (such as gearbox), (2) performs dynamic state estimation (such as selected gear and brake cylinder pressure at $t = 0$) and tracking (latent state variable values for $t > 0$), and (3) successfully performs system identification and parameter estimation (such as the aerodynamic drag effect and its coefficient).

This chapter focuses on outlining the process of applying deep learning and deep reinforcement learning for modeling and control of heavy duty trucks. More extensive experimentation and comparison with established classical approaches is still required for validation and performance evaluation. Furthermore, the process presented here still requires full replication for each target truck, and each truck combination (multi-truck environments). Further investigation is still required to introduce transfer learning of longitudinal dynamics across mechanical configurations. Data sampling efficiency and utilization of existing first-principle models could also be investigated to improve the process presented here.

Chapter 4

Vehicle actuation using deep reinforcement learning

4.1 Introduction

Cruise control is considered, as of today, one of the fundamental driver assistant systems for freeway driving. Previous chapter presented and validated a process, summarized in Figure 3.1, for modeling and control of heavy duty trucks using deep learning and deep reinforcement learning (deep-RL). Starting with the same pre-trained deep truck models and following the same process, this chapter composes and studies deep-reinforcement-learning-based longitudinal cruise control systems for heavy duty trucks. The chapter uses deep neural networks as a model for the controller module and trains it using deep reinforcement learning framework. The deep reinforcement learning environment model is embedded with deep truck models to reduce simulation-to-reality-gap and to study controller transfer-ability from offline deep training environment to real physical systems. The chapter evaluates transfer and robustness performance, and validate control accuracy for heavy duty trucks of different configurations both in simulation (TruckSIM truck) and using real-physical trucks (Freightliner and Volvo trucks) using a variety of operational scopes and scenarios.

4.2 Problem formulation

In this chapter, we use deep-reinforcement-learning to design end-to-end heavy duty truck controllers allowing for offline (1) design/tune the cruise controller, (2) calibrate the controller module to the specific physics of each truck, (3) design an embedded state observer/tracker. In developing these controls we assume limited observable IO, unknown truck mechanical configuration, and unknown relevant internal state. We formulate the problem as a *Partially Observable Markov Decision Processes* (POMDP) as presented in previous chapter and solve it using deep reinforcement learning framework.

Deep-RL longitudinal heavy duty truck cruise control

This section presents cruise control as a deep-RL continuous control task. The deep-RL environment transition probability distribution P is implemented using deep-truck non-linear dynamics model defined previously in Equation 3.1 in Chapter 3 as follows:

$$\begin{bmatrix} x(k+1|\Phi_{\text{DT}}) \\ y(k+1|\Phi_{\text{DT}}) \end{bmatrix} = f_{DT}\left(\begin{bmatrix} u(k) \\ w(k) \end{bmatrix} \middle| \begin{bmatrix} x(k|\Phi_{\text{DT}}) \\ y(k) \end{bmatrix}, \Phi_{\text{DT}} \right),$$

where x represent internal state of the truck, y represent truck response, u represent controllable inputs to the truck, w represent uncontrollable conditions relevant to the dynamics, Φ_{DT} represent internal truck model parameters, and k is discrete time-step of size dt. The model f_{DT} is a recurrent deep learning model for which internal state encoding is dependent on Φ_{DT}.

The agent is represented by the probability distribution function $\pi(a_k|o_k, \Phi_{\text{agent}})$ and the corresponding control $u(k)$ is implemented as:

$$\begin{aligned} u(k) &= f_\pi\left(\begin{bmatrix} y(k) \\ y_{\text{target}}(k) \\ w(k) \end{bmatrix} \right) \\ &= E\left(\pi\left(a_k \middle| o_k = \begin{bmatrix} y(k) \\ y_{\text{target}}(k) \\ w(k) \end{bmatrix}, \Phi_{\text{agent}} \right) \right) \end{aligned}$$

representing the mean value for a Multi-Layer Perceptron (MLP) Gaussian distribution model.

The agent is rewarded for regulating truck response y at target response y_{target} as follows:

$$r(t) = \alpha \left\| y(k) - y_{\text{target}}(k) \right\|_2^2 + \beta \left\| u(k) \right\|_2^2,$$

where α and β are positive constants.

Initial distribution of the environment ρ_o is given by the initial distributions of the set $\{x, y, w, y_{\text{target}}\}$. The internal state of the truck x is initialized to zero as specified in Chapter 3. The variables y, w are initialized by sampling from the maximum ignorance uniform distribution $\text{uniform}(\min, \max)$ where the bounds of each distribution are specified according to the expected operating range of the controller and the validity domain of the deep truck models. The variable y_{target} (modeled as a constant during training) is initialized by sampling from $y_o + \text{uniform}(-y_\delta, y_\delta)$ where distribution of initial regulation error $|y(k=0) - y_{\text{target}}(k=0)|$ comes from maximum ignorance uniform distribution bounded by y_δ.

Variable instantiations are detailed for each respective experiment in the later sections; however, in general, for a cruise control,

$$\begin{aligned} u(k) &= \begin{bmatrix} E_{\text{cmd}}(k) \\ B_{\text{cmd}}(k) \end{bmatrix} \\ y(k) &= v(k) \\ y_{\text{target}} &= v_{\text{target}}(k) \\ w(k) &= \theta_{\text{rdg}}(k), \end{aligned}$$

where $E_{\text{cmd}}(k)$ is engine command, $B_{\text{cmd}}(k)$ is brake command, $v(t)$ is vehicle speed, $v_{\text{target}}(k)$ is target speed (set-value, or set-speed), and $\theta_{\text{rdg}}(k)$ is road grade each at discrete time step k.

Training and transfer procedure

The deep-RL agent policies are trained offline (with respect to target truck) simulated environment using the model free policy gradient TRPO [84], [86] algorithm. For validation, agent policies are zero-shot transferred for closed loop evaluation (online) on each truck. To enable transfer, we minimizing model mismatch and close the reality gap by embedded high accuracy models of each truck of interest using Deep Truck Model approach in Chapter 3.

4.3 Experiments

This section presents experimental evaluation of the control approach presented in this chapter. We conduct experiments in simulation for detailed evaluation and replicate the procedure using two full-size real-physical trucks that have different mechanical configurations. Control performance is validated against the model based controller presented in Section 2.2 (Cruise Control).

Experiment name/label notation

For convenience of cross-referencing, in this chapter, we label each experiment using a cascade of codes ‘code/1-code/2-code/3-. . . ’ and use the following codes:

Code/1: Experiment truck platform/environment code. This code references the truck/environment used to conduct the experiment. The code takes one of the following variations:

DE: This code references experiments conducted using deep environment (deep truck model based).

TS: This code references experiments conducted using TruckSIM.

FL: This code references experiments conducted using real full-size Freightliner truck.

VL: This code references experiments conducted using real full-size Volvo truck.

Code/2: Target velocity waveform code. This code references the target velocity (or set-speed) waveform used to conduct the experiment. The code takes one of the following variations:

STEP: This code references experiments conducted using a step function.

LRMP: This code references experiments conducted using a low acceleration ($|[0-0.5]|$ m/s^2) ramp function.

HRMP: This code references experiments conducted using a high acceleration ($|[1-1.5]|$ m/s^2) ramp function.

SINE: This code references experiments conducted using a sine wave.

Code/3: Road topography code. This code references the major topography features of the road used to conduct the experiment. The code takes one of the following variations:

FLT: This code references experiments conducted over a flat (or nearly flat) road.

NFLT: This code references experiments conducted using a graded (non-flat) road.

Code/4: Control policy identification code. This code is used to help identify the control policy used for each experiment. The code takes one of the following variations:

DTSDP: This code references experiments conducted using a deep policy trained based on a deep model of the TruckSIM truck platform.

DFLDP: This code references experiments conducted using a deep policy trained based on a deep model of the Freightliner truck platform.

DVLDP: This code references experiments conducted using a deep policy trained based on a deep model of the Volvo truck platform.

TSMBH: This code references experiments conducted using the model based hierarchical controller presented in Section 2.2 (Cruise Control).

Code/5: Control policy note code. This code references other notes relevant to the policy or the experiment. The code takes one of the following variations:

MG: This code indicates that the control policy models gravity (this is a default for the model based controller).

NMG: This code indicates that the control policy does not model gravity (this is a default for the deep policies).

EOB: This code indicate that the control policy is operated outside of the engine command training distribution bound.

Numerical validation experiments

Simulation environment and truck mechanical configuration

The experiments in this section are conducted in TruckSIM [74], a black-box state-of-the-art commercial software framework with high fidelity modeling capabilities and a detailed vehicle and vehicle component libraries.

The truck, shown in Figure 3.3, is equipped with a 402hp engine. The engine shaft is connected to one side of the transmission via clutch. The clutch allows speed difference between the engine and the transmission when gear shifts. The transmission has ten forward gears and one reverse gear. The other side of the transmission is connected to rear wheels via a differential gear with a fixed reduction ratio. The truck is equipped with an air-brake system. The front air-brakes have capacity of 7.5 kN-m on each wheel. The rear brakes have capacity of 10 kN-m on each wheel. Actuation control input to the truck are engine torque and brake cylinder pressure. The details are presented here for completeness and for reporting purposes, but are irrelevant to the deep model.

Training setup and learning curves

The deep-RL environment is embedded with a deep truck model learnt for the truck described in this section using open loop driving data as described in Chapter 3. For this experiment, the sampling rate is set to $10Hz$ ($dt = 0.1s$). The truck is not equipped with any sensors relevant to the driving environment (e.g. road grade) and thus we substitute $w(k)$ with the empty set. The controllable input to the truck (agent output) is given by $u(k) = [E_{\text{cmd}}(k), B_{\text{cmd}}(k)]$, where $E_{\text{cmd}}(k)$ is requested engine torque in $[N-m]$ and $B_{\text{cmd}}(k)$ is requested service brake master cylinder pressure percentage $[0-100\%]$.

Agent observation vector is given by $o(k) = [v(k), v_{\text{target}}(k), \theta_{\text{rdg}}(k)]$, where $v(k)$ is longitudinal speed in $[m/s]$, $v_{\text{target}}(k)$ is target longitudinal speed in $[m/s]$, and $\theta_{\text{rdg}}(k) = 0$ for a flat road used for this experiment. The agent π is modeled using an ANN that has 3 hidden layers, each of size 25.

The deep-RL controller was trained on RLLab [84] using batch size of 20000, max path length of 800 (sampled at 10Hz) and discount factor of 0.9999. We trained ten policies (ten seeds). The average discounted returns plot is shown in Figure 4.1. Training of all ten policies started with random agent initialization and hence the low initial return. The first 100 iterations are exploration intensive leading to high variance between trained policies. Variance between these policies reduces and converge after 200 iteration.

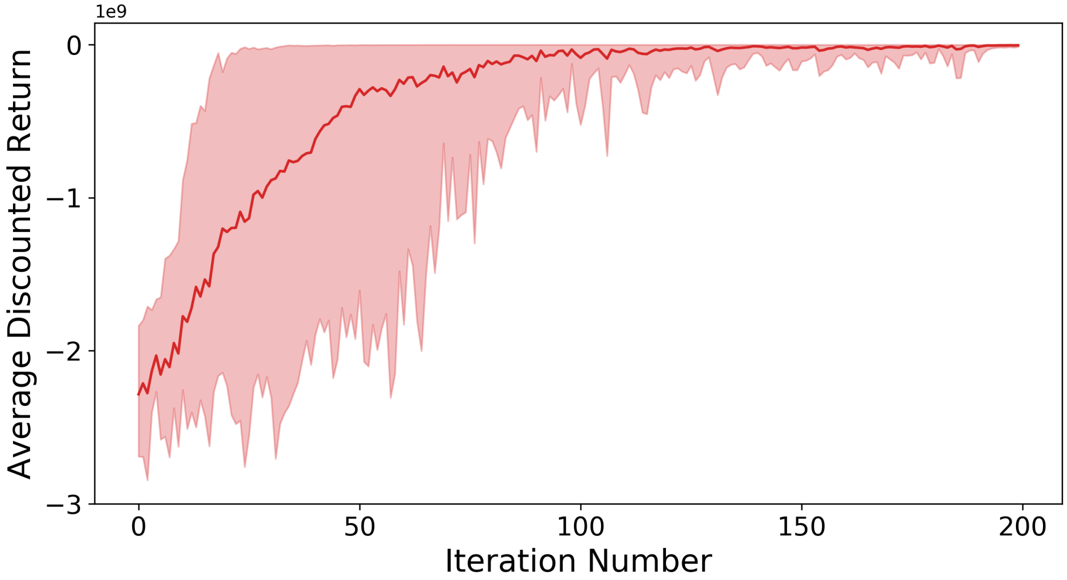


Figure 4.1: Learning curve—min/max (light red shaded area) and mean (dark red curve) from ten seeds—for deep cruise control policy based on deep model of the TruckSIM truck.

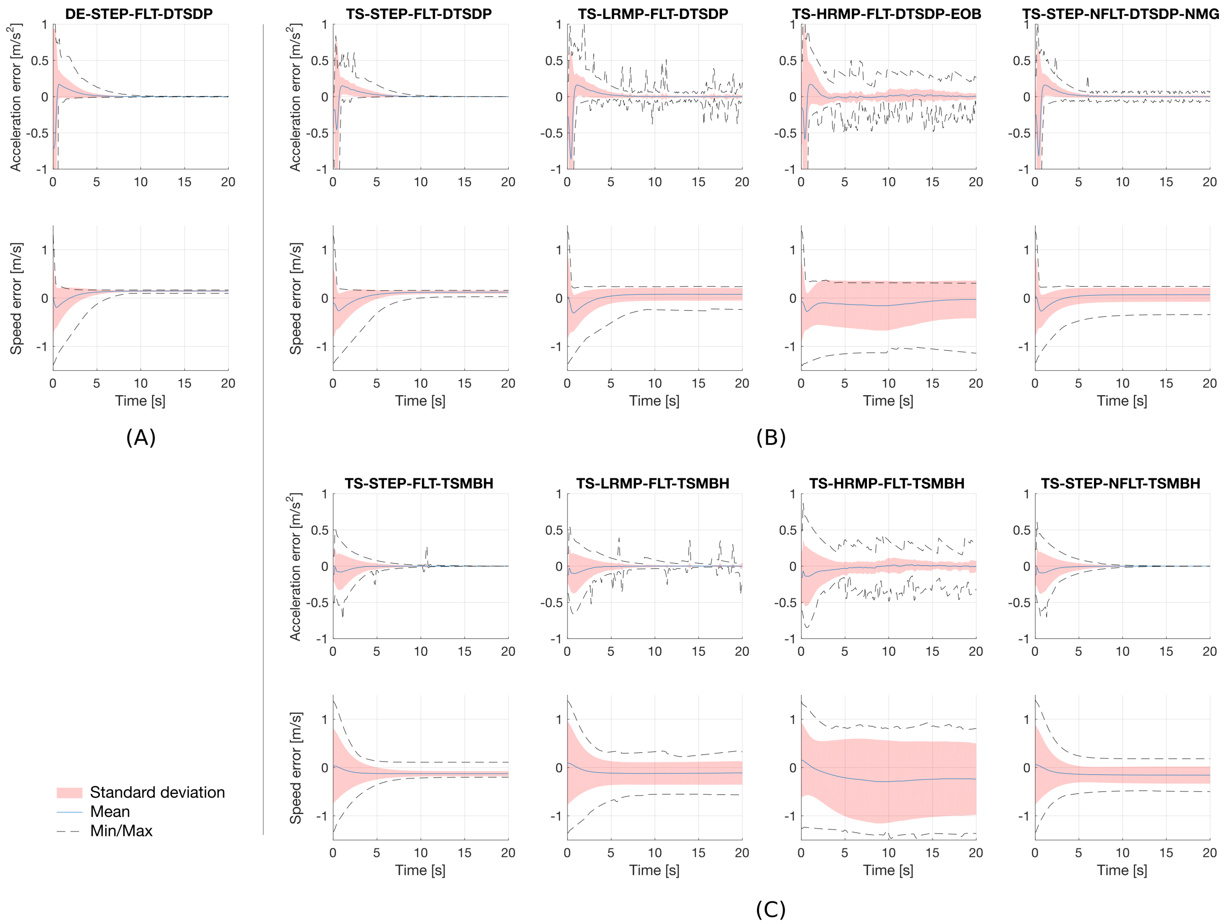
Control validation setup and validation results: Control performance, transfer, and sensitivity

This section validates the performance of the deep-RL cruise controller developed in the sections above, evaluates its transfer performance, and sensitivity to several of the major assumptions used to train the controller. Deep-RL controller performance is compared to a full-featured nonlinear classical model-based controller baseline. Compared to the bare-minimum deep-RL controller, the baseline controller is calibrated using ground truth parameters (TruckSIM model parameters), real-time truck internal-state feedback (gear ratio), and gravity model.

Figure 4.2 shows closed loop experimental performance statistics conducted against (A) the deep truck environment, (B) TruckSIM environment, and (C) replicates the TruckSIM environment set of experiments using the baseline classical controller.

DeepEnv-set experiment is a replication of the training setup and consists of 100 rollouts drawn from the same training distributions (environment model, initial speed, set-values, etc). The same controller is zero-shot transferred to TruckSIM to produce TruckSIM-set consisting of 100 rollouts drawn from the same initial speed and set value distributions.

TS—LRMP—FLT—DTSDP, *TS—HRMP—FLT—DTSDP—EOB*, and *TS—STEP—NFLT—DTSDP—NMG* experiment sets assess controller performance sensitivity to the major assumptions. The deep-controller was trained assuming constant set-speeds for each rollout. To violate this assumption, in *TS—LRMP—FLT—DTSDP*,



(A)

(B)

(C)

Figure 4.2: Control error statistics generated from 100 random trails per experiment set for (A) the deep policy evaluated against the deep environment, (B) the deep policy evaluated against TruckSIM, and (C) the model based hierarchical controller.

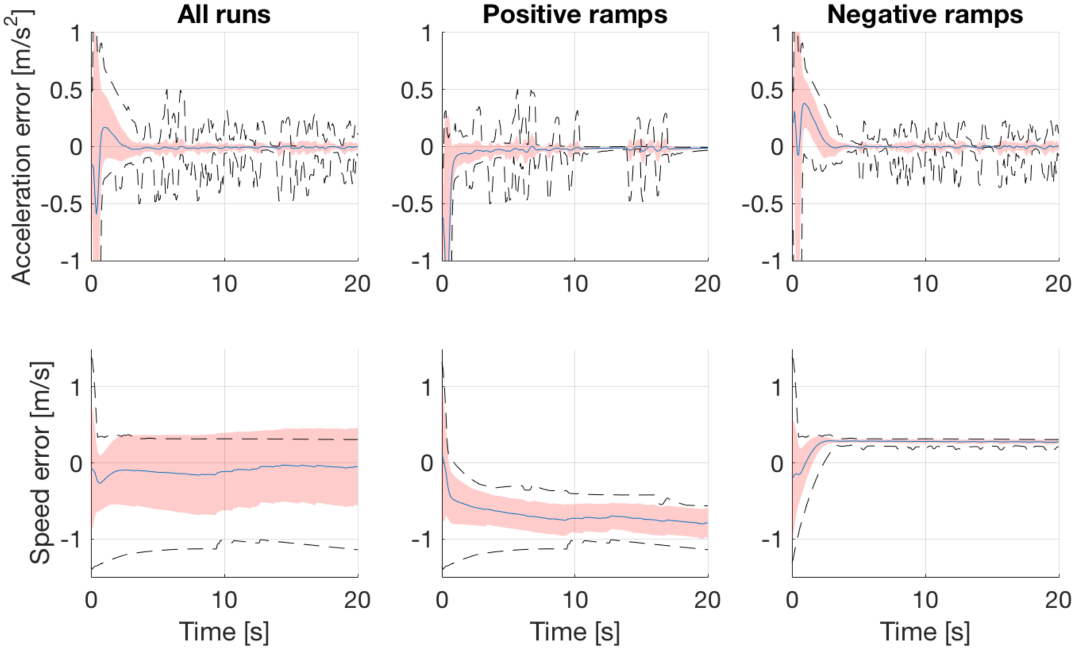


Figure 4.3: A decomposition of validation results—min/max (dashed curves), standard deviation (red shaded areas) and mean (blue curves)—of *TS—HRMP—FLT—DTSDP—EOB* experiment set into acceleration ramps and deceleration ramps.

100 rollouts were conducted using ramp set-speed trajectories drawn from $v_{\text{target}}(k) = v_{\text{target}}(0) + a_{\text{LRMP}}k$, where $v_{\text{target}}(0) - v(0)$ is drawn from uniform$(-1.39, 1.39)[m/s]$ and a_{LRMP} is drawn from uniform$(-0.5, 0.5)[m/s^2]$.

Training distributions were specified based on observation space; i.e., to cover/span operational-distributions for initial-speeds and initial-errors, and implicitly assume these distributions is action-space spanning; it spans corresponding operational-distributions of engine and brake command spaces. The *TS—HRMP—FLT—DTSDP—EOB* experiment-set explicitly violate this assumption in addition to the constant set speed assumption. During training, action space was limited to $0 - 1000N - m$ and then operated in truckSIM for $0 - 1700N - m$. To explicitly operate the engine torque out of bound, the v_{target} trajectory distribution was drawn from ramps with high accelerations; a_{HRMP} is drawn from a uniform distribution supported by $(-1.5, -1.0)$ and $(1.0, 1.5)$ $[m/s^2]$.

The training environment assumed flat roads. To violate this assumption, we replicated the TruckSIM-set on non-flat roads. Road grade is assumed constant for each rollout and is drawn from -2% to 2% grid spaced by 0.25%.

For all TruckSIM environment experiments, after the agent is trained offline in DeepEnv, it is transferred (unchanged) and tested online in TruckSIM against the same TruckSIM truck used for model development. The simulation environment is setup with a single truck

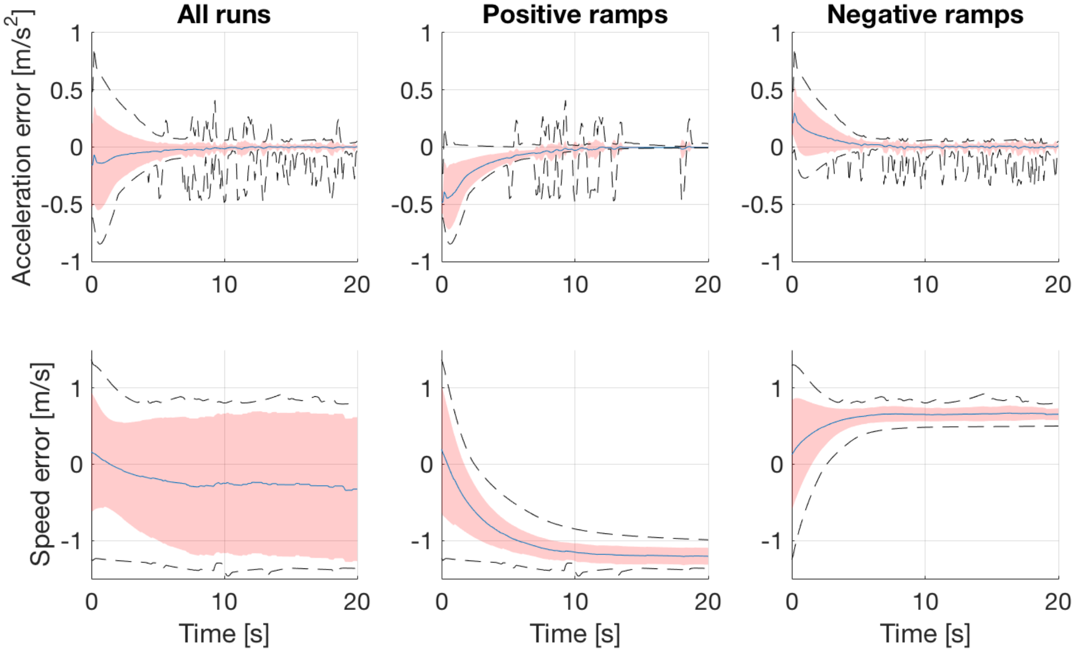


Figure 4.4: A decomposition of validation results—min/max (dashed curves), standard deviation (red shaded areas) and mean (blue curves)—of *TS—HRMP—FLT—TSMBH* experiment set into acceleration ramps and deceleration ramps.

controlled at 10Hz on a flat road (unless otherwise is stated).

In TruckSIM, to automate the simulation process and ensure consistency, to initialize each rollout, we drive the truck to $v_{\text{target}}(0) + 0.28m/s$ then apply constant break cylinder pressure of 0.1 until vehicle speed v is within $0.1m/s$ from $v_{\text{target}}(0)$ then activate experiment session and activate the controller. Each experiment (rollout) is run for 80s (800 steps).

For each experiment set, Figure 4.2 shows mean, variance, and min/max of speed error $e_v(k) = v(k) - v_{\text{target}}(k)$, and shows mean variance, and min/max of acceleration error $e_a(k) = \dot{e}_v(k)$.

All experiments are initialized with speed error drawn from uniform$(-0.5, 0.5)[m/s]$. Experimental results show stability of the deep-RL cruise controller for which acceleration errors converge to near zero for all experiments and all experiment sets. Acceleration error converges to zero, and hence cruise control steady state speed is reached, within $5s$.

Speed error statistics for *TS—HRMP—FLT—DTSDP—EOB* appear significantly larger compared to the other experiment sets. In this set, higher accelerations magnify the effects of truck control delays. Speed error caused by delay creates a bimodal distribution consistent with the bimodal support of v_{target} distribution (positive error for negative accelerations and vice versa) as can be seen in Figure 4.3 and Figure 4.4.

Transfer performance and sensitivity is primarily seen from the speed steady state error distributions. Transfer from DeepEnv to TruckSIM-set shifts the speed steady state mean from $0.145m/s$ to $0.121m/s$ and the variance from 0.017 to 0.033. Violation of the constant set speed assumption shifted the mean to $0.077m/s$ and the variance to 0.129. The combined violation of constant set speed assumption and engine torque out of bound shifts mean to $-0.026m/s$ and the variance to 0.381. Driving over non-flat roads shifts the mean to $0.069m/s$ and the variance to 0.146.

Figure 4.2 shows stability of the learned deep-RL cruise controller, shows transfer of the controller from the deep-truck environment to the TruckSIM environment with relatively marginal shift in error statistics, and its tolerance to some shifts in distributions from those of the training environment with reasonable error trade-off. The base control scenarios (*STEP—FLT*) primarily suffer from an over-estimation of mechanical resistances leading to positive constant steady state error. Steady state error variance increases for *LRMP* and *HRMP* due to the additional delay error, not observable for constant target speeds, and increases proportionally with acceleration. Steady state error variance in *NFLT* experiment-sets is affected by the change in dynamics due to effective gravitational force. Performances and tolerances (shifts in distributions) are consistent with those from the baseline controller.

Field experiments

This section presents field experimental results for the controllers described in this chapter. The section documents experiments conducted using two differently configured real-physical heavy duty trucks. These same two trucks were modeled using two different physics-based power-train models in [14] and [15] used to develop high precision control systems for each respective article.

Truck platforms/configurations and field driving environment

This section gives an overview of the full-size trucks, shown in Figure 3.3 used to conduct the experiments in this chapter.

Freightliner truck. The Freightliner truck used for the results in this section is a tractor-only Freightliner Century truck driven by a 435 hp turbocharged Detroit Diesel diesel engine and equipped with a 6 gear true-automatic (equipped with torque-converter) Allison transmission system. The service brake is a drive by wire all the way to the wheels. The truck is not equipped with road grade sensors. Experiments conducted using this truck has been carried out at a nearly flat test track with straight roads the longest of which is around 300 meters long at the Richmond Field Station at California.

Volvo truck. The second set of experiments were conducted using a tractor-only Volvo VNL truck driven by a 500 hp engine with an automated manual-transmission (equipped with clutches). Experiments conducted using this truck has been carried out primarily at open non-flat freeways.

Truck interface for control

The vehicles used for experiments in this section were developed for automated vehicle research and fitted with safety installations appropriate for automation and control experiments. Actuation, feedback, and data logging utilizes direct access to the standard vehicle network J1939-bus at 10Hz.

Validation results and discussion

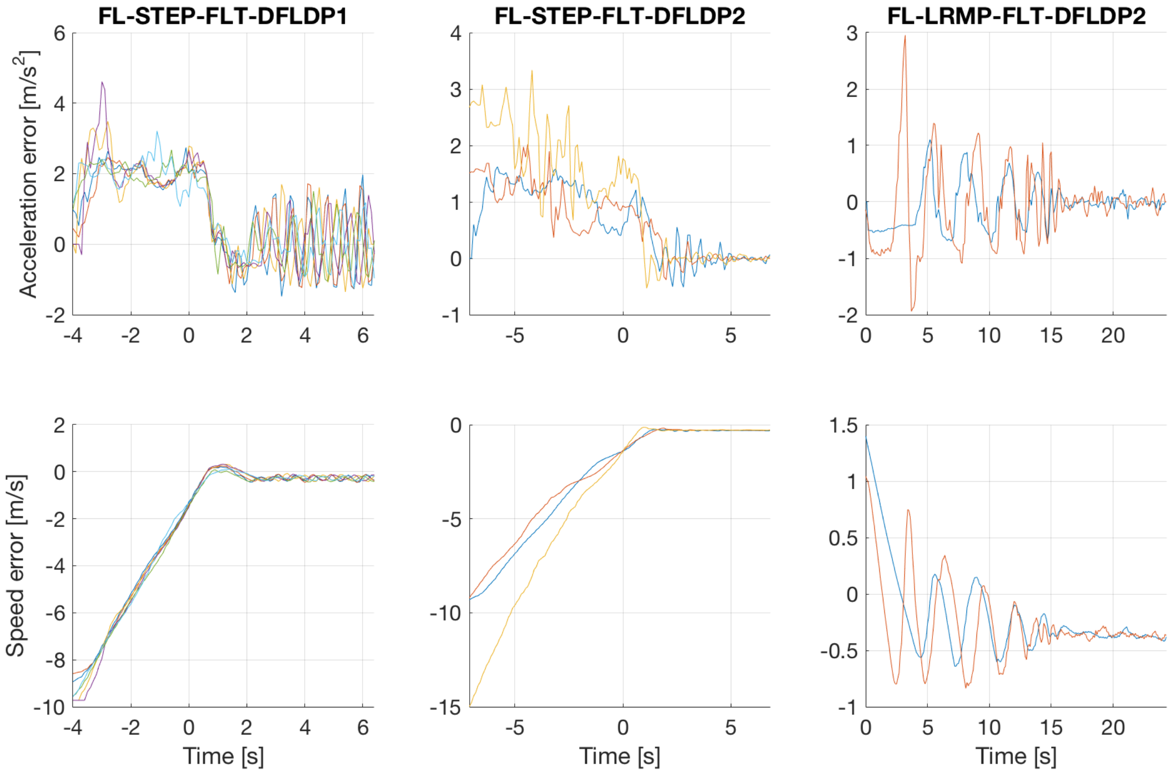


Figure 4.5: Control error signals for experiment sets using the Freightliner truck.

This section presents field validation results for the deep-RL cruise controller developed for full size real-physical heavy duty trucks. We conduct six sets of experiments to validate transfer and controllers' sensitivity to major assumptions employed during training. We conduct these sets of experiments using two different trucks, three different controllers, and over flat and graded roads. Error signals and error statistics for the conducted experiments are shown in Figure 4.5, Figure 4.6, Figure 4.7, and Figure 4.8.

The *FL—STEP—FLT—DFLDP1* experiment set consists of step function set-speed control trajectories conducted using the FL on a nearly flat road. This control policy is one of

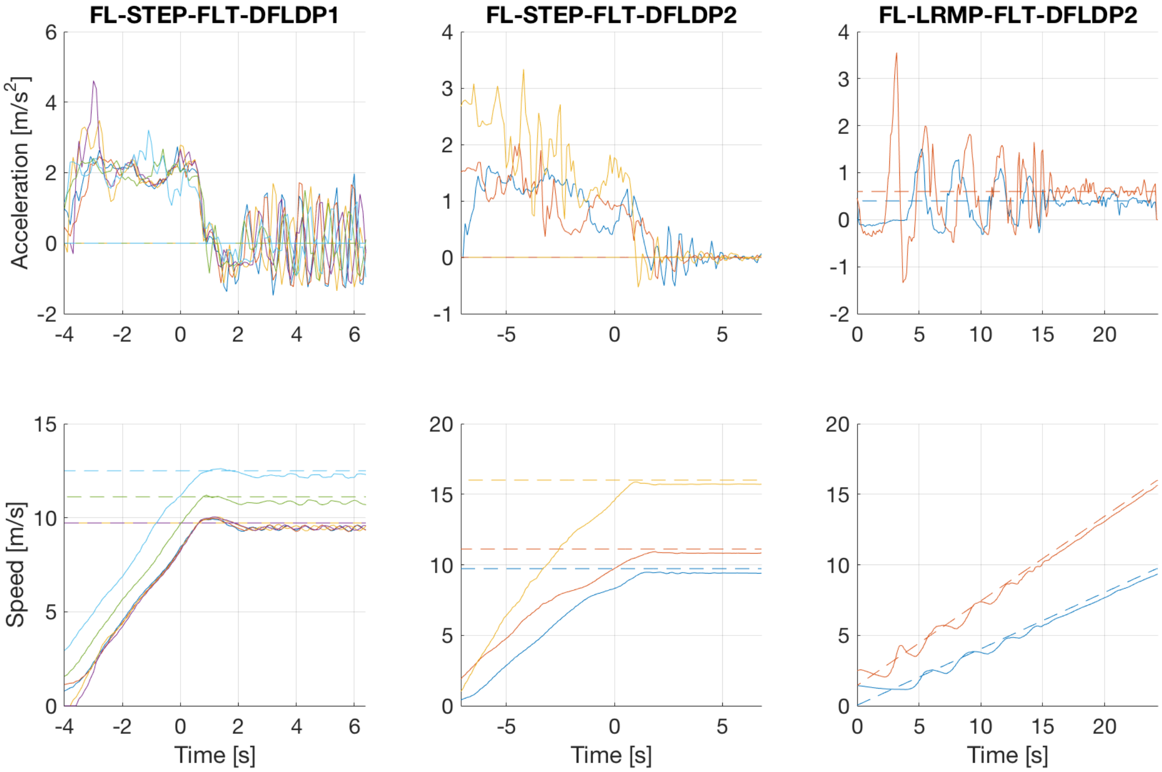


Figure 4.6: Speed and acceleration signals for experiment sets using the Freightliner truck—set value signals in dashed lines, and truck response in solid lines.

two policies we used for validation in this section. We conducted six rollouts all with near zero initial speed; initial speeds and initial errors are outside training bound. A set value of 9.72m/s was used for four of the rollouts, and a set value of 11.11m/s and 12.5m/s were used for one rollout each. Rollout results of this experiment were aligned (along time axis) at the boundary of the training initial error distribution (1.39m/s).

This controller appears to be underdamped with bounded steady state oscillations. Steady state oscillations are consistently bounded between -0.46 and -0.069 m/s with overshoot of 0.32 m/s from set value.

The *FL—STEP—FLT—DFLDP2* experiment set consists of step function set-speed control trajectories conducted using the FL on a nearly flat road. This set was conducted using a second policy. We conducted three rollouts all with near zero initial speed; initial speeds and initial errors are outside training bound. A set value of 9.72 m/s, 11.11m/s, and 16m/s were used for one rollout each. The rollouts of this experiment were aligned (along time axis) at the boundary of the training initial error distribution (1.39m/s).

Accelerations converged to within steady state error of ±$0.2 m/s^2$ for this set of experi-

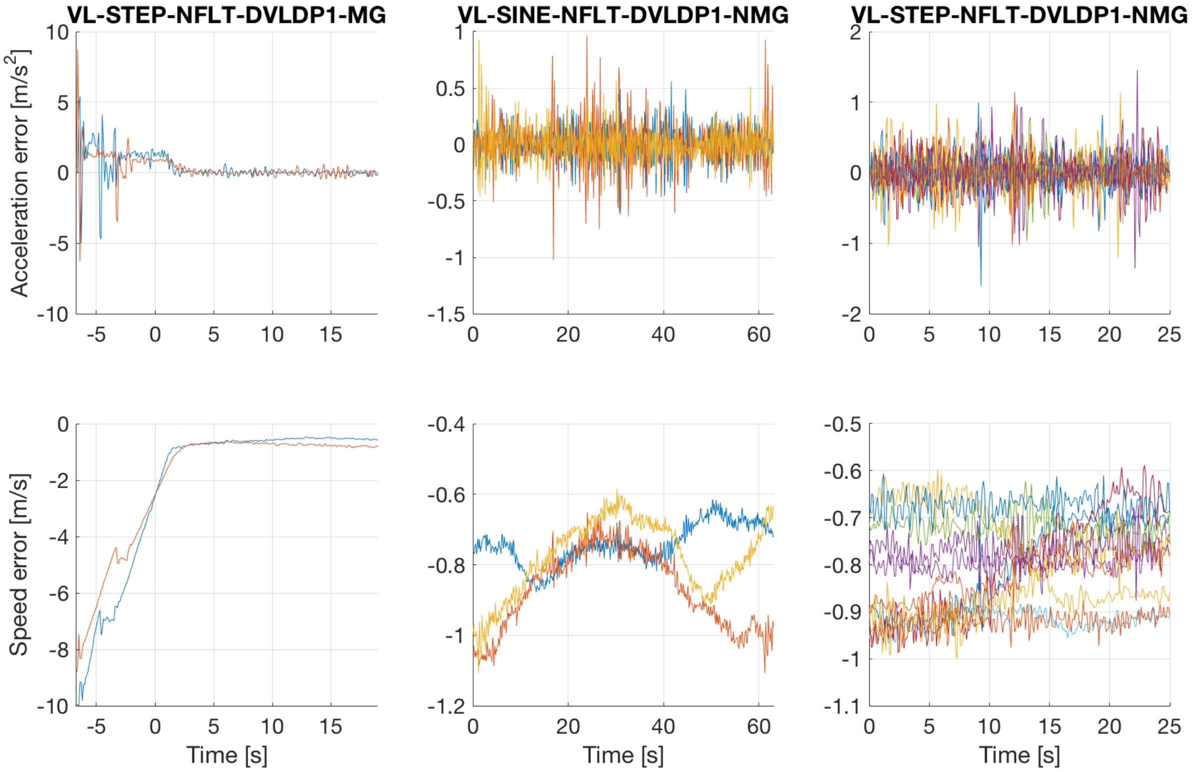


Figure 4.7: Control error signals for experiment sets using the Volvo truck.

ments resulting in a smoother stable steady state speeds. Steady state speed converged to within -0.34 and -0.26 m/s.

The *VL—STEP—NFLT—DVLDP1—MG* experiment set consists of step function set-speed control trajectories conducted using the Volvo on a non-flat road using a controller policy with road grade feedback. We conducted two rollouts; the first rollout with an initial speed of 10m/s and set speed of 20m/s, and the second rollout with initial speed of 16m/s and set speed of 25m/s. The rollouts of this experiment were aligned (along time axis) at the boundary of the training initial error distribution (2.5m/s).

Accelerations converged to within steady state error of $\pm 0.62 m/s^2$ for this set of experiments resulting in a smoother stable steady state speeds. Steady state speed converged to within -0.85 and -0.48 m/s. Drops in speed at -4.5s and -3.2s are gear-shifting instances.

The *FL—LRMP—FLT—DFLDP2* experiment set consists of ramp function set-speed control trajectories conducted using the FL on a nearly flat road. We conducted two rollouts all with near zero initial speed; initial speeds and initial errors are outside training bound. The (constant) acceleration of the first ramp function is $0.4 m/s^2$ and the second ramp function is $0.6 m/s^2$.

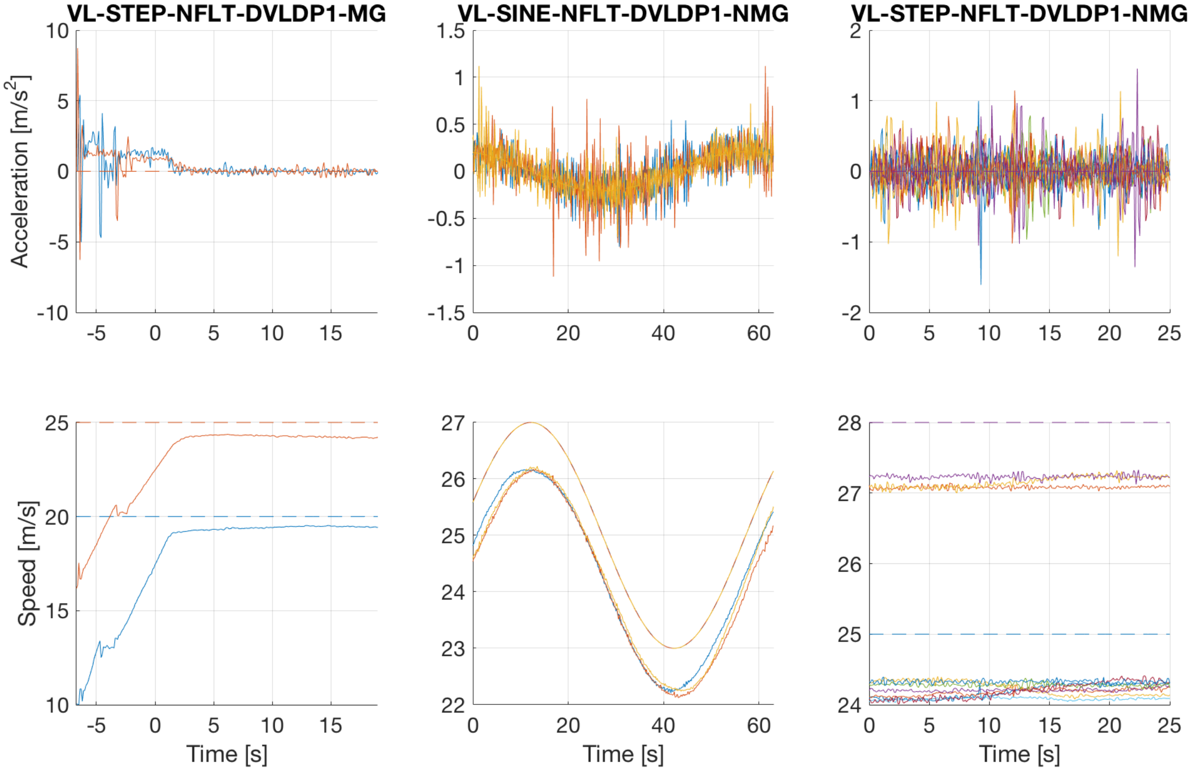


Figure 4.8: Speed and acceleration signals for experiment sets using the Volvo truck—set value signals in dashed lines, and truck response in solid lines.

Accelerations converged to within steady state error of $\pm 0.3 m/s^2$ and steady state speed converged to within -0.42 and -0.28 m/s. Decaying transient oscillations were observed with maximum error of -0.80 and 0.75 m/s after initial error and before convergence to steady state error. These oscillations were primarily caused by the still disengaged torque converter, which engages at about $8 \sim 10 m/s$. These oscillations do not appear in the step response because of the higher acceleration requested in those experiments.

The *VL—SINE—NFLT—DVLDP1—NMG* experiment set consists of sine function set-speed control trajectories conducted using the Volvo on a non-flat road using a controller policy with no road grade feedback. We conducted three sine cycle runs with average speed of 25m/s, amplitude of 2 m/s, and cycle of 60 s.

Speed of this set of experiments converged to within -1.1 m/s and -0.6 m/s. The higher speed error in this experiment is caused by the un-controlled for gravity effects. Speed and acceleration of this experiment have higher amplitude white noise than the other experiments.

The *VL—STEP—NFLT—DVLDP1—NMG* experiment set consists of constant function set-speed control trajectories conducted using the Volvo on a non-flat road using a controller policy with no road grade feedback. We show the steady state ensemble of eleven 25-second rollouts.

Speed of this set of experiments converged to within -1.0 and -0.6. The higher speed error in this experiment is caused by the un-controlled for gravity effects. Speed and acceleration of this experiment have higher amplitude white noise than the other experiments.

In this section, transfer performance from DeepEnv of two different policies to the same real-physical truck, and transfer of two policies to two different trucks were shown. Moreover, control performance sensitivity to violating the constant set speed assumption and to road profile modeling were assessed. The controllers achieved consistent performance across a wide range of the state space of the truck with minimum feedback information for the set of conducted experiments.

Deep control performance compared to baseline

The deep-RL based feedback controller is tested in the field against the same real truck. Control experiments here were limited to engine drive control. The scenarios were constrained by the relatively short length of the test track. Figures 4.9, 4.10, and 4.11 show experimental control performance results for a step and a saturated ramp functions. These experiments demonstrate comparable steady state control performances between RL and model-based control. The offset between reference value and RL-control response for both experiments, and the larger transient oscillations for the ramp function experiment are artifacts of the optimization-based design cost function we used. Namely, the offset is an artifact of the trade-off between input cost (non-zero engine torque to counter resistances) and speed error cost. The larger transient oscillations are artifact of the cost function's emphases on accurate speed tracking, which resulted in an aggressive high gain controller.

4.4 Conclusion

This chapter presented preliminary experimental evaluation of deep-RL-based longitudinal heavy duty truck cruise control. The experiments demonstrated transfer (from offline model to online real environment) and robustness of controllers learned offline based on deep truck model environment. The experiments evaluated the potential of the pipeline (deep truck and deep-RL) as configuration agnostic strategy to develop non-linear controllers for heavy duty trucks.

The deep model and deep-RL controls presented in this chapter successfully (1) infers relevant latent and state variables (such as gearbox), (2) performs dynamic state estimation (such as selected gear and brake cylinder pressure at $t = 0$) and tracking (latent state variable values for $t > 0$), and (3) successfully performs system identification and parameter estimation (such as the aerodynamic drag effect and its coefficient).

Deep learning and deep RL offer a convenient alternative to modeling and control of dynamical system. This convenience however comes at the expense of formal guarantees

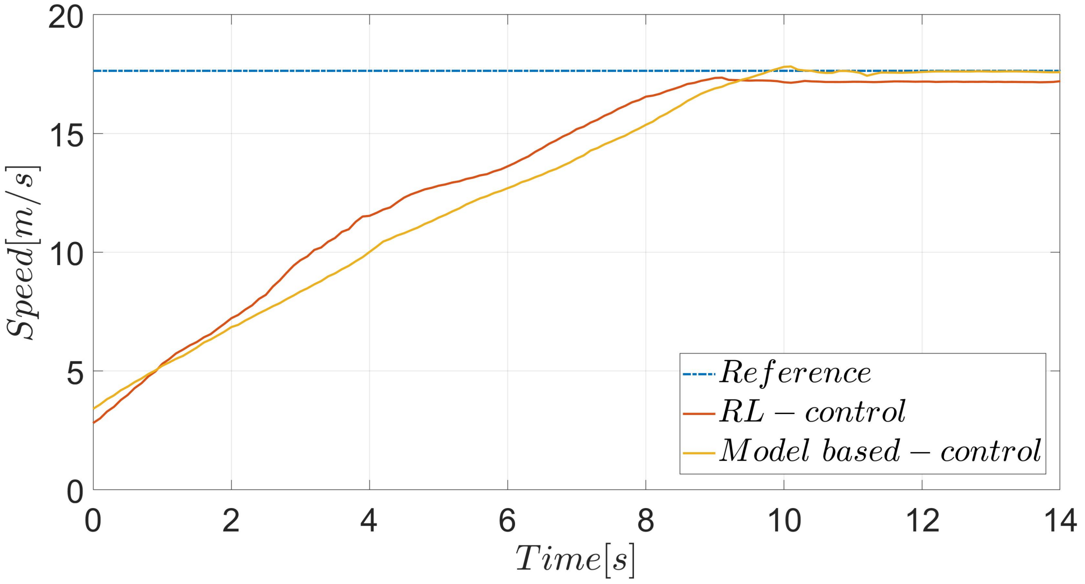


Figure 4.9: Control performance for $17.63m/s$ step set value function conducted using Freightliner truck.

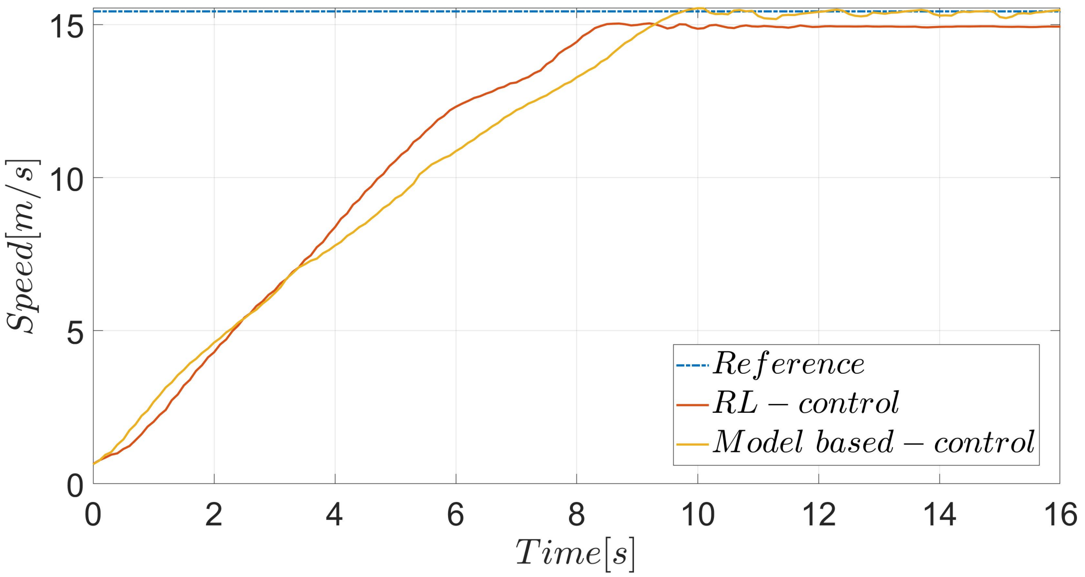


Figure 4.10: Control performance for $15.43m/s$ step set value function conducted using Freightliner truck.

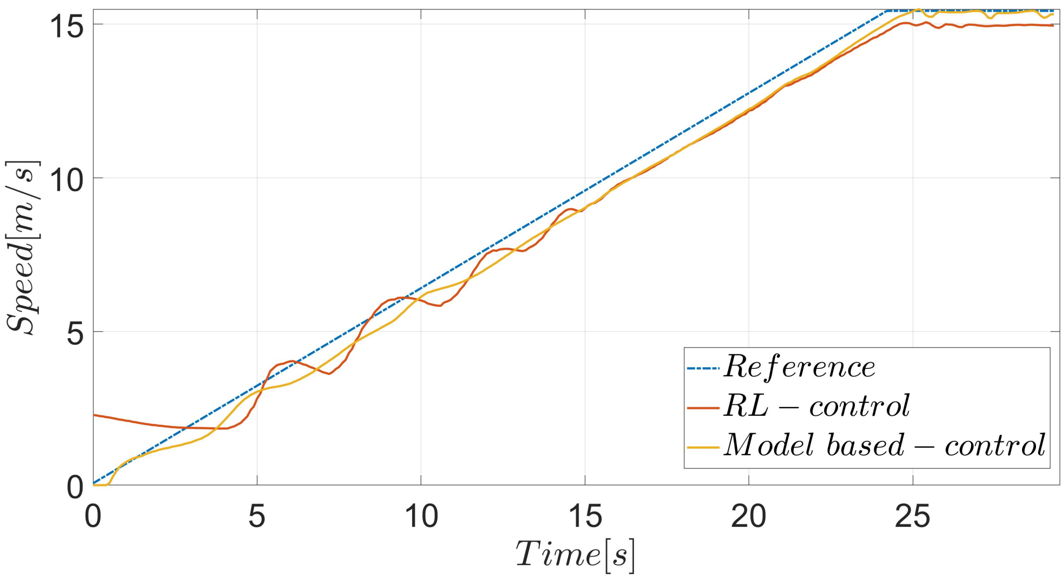


Figure 4.11: Control performance for $15.43 m/s$ saturated ramp set value function conducted using Freightliner truck.

requiring more exhaustive evaluation of possible variations such as vehicles and operational details, and a more extensive field testing is still needed. While safety is outside the scope of this chapter, failure conditions are expected to occur and must be studied and accounted for. Moreover, the preliminary transfer and robustness performance results presented here must be studied and evaluated in more details in relationship to the choice of deep learning and deep-reinforcement learning algorithms and models, to the expected performance of the source model used to develop the deep environments, and more importantly in comparison to policies (1) learnt directly using the real-environment, and (2) learnt using classical vehicle models.

Chapter 5

Multi-scale vehicle and traffic flow modeling and actuation using mean-field models

5.1 Introduction

The earlier chapters of this manuscript investigated the automation of heavy duty trucks. Namely—in Chapters 2, 3, and 4—modeling and control of heavy duty trucks using classical methods and using machine learning have been investigated, in addition to high fidelity simulation and field development and implementation. Section 3.5 of Chapter 3 briefly investigated microscopic modeling and control of multi-truck systems. This chapter extends these notions to investigate the automation of the heavy duty truck within the bigger context of a vehicular traffic stream on the road.

Vehicular traffic on the road can be inherently unstable as has been experimentally demonstrated in [89], [90]. The presence of automated vehicles within this traffic stream can influence these streams with potential for capacity multiplication [18] or creating bottlenecks [63], [64]. In [85], [91], microscopic numerical and field experimental studies have been conducted for the automated vehicle as a traffic flow smoothing agent to improve capacity and energy consumption in the simplified setting of ring roads.

Demonstrating these gains in open networks such as an open freeway require more detailed models and controllers, an effort being investigated by the Circles project with a multi-campus team lead by UC Berkeley. Flow [85], [92] proposes deep reinforcement learning framework to study microscopic traffic flow instabilities and their control. Transfer of traffic flow smoothing controllers controllers from ring roads to open networks has been investigated in [93] and from simulation to scaled physical models has been studied in [94]. However, computational cost of microscopic simulation of traffic scales with the number of simulated vehicles, which can be expensive for open networks.

Macroscopic models [95]–[98] have historically been considered an effective tool to study large scale traffic systems. Mean-field modeling framework provides an approach to derive macroscopic models given descriptions of the microscopic interactions between considered agents [99]–[101]. These mean-field models allows for the investigation of such traffic streams at multiple scales. Namely, it provides a formal framework to study microscopic automated vehicles as they drive within a traffic stream that is modeled macroscopically, and allows for the development of optimal control systems to actuate this traffic.

The mean-field limit to model such system yields models representing PDE nonlocal conservation laws. The literature on numerical solvers for these models is still sparse. Namely, to the best of our knowledge, [102] is the only published numerical solver. This chapter focuses on developing a structure preserving numerical solver algorithm suitable to study multi-scale traffic models and flow actuation.

5.2 Nonlocal conservation laws and mean-field models

Conservation laws [103] have been used to describe a wide range of phenomena such as conservation of energy, conservation of mass, and conservation of momentum. Many other phsyical phenomena are naturally nonlcal however motivating the theory of nonlocal conservation laws. These laws arise from mean field limits to describe interactions such as pedestrian dynamics, swarms, and traffic flow. The nonlocal theory is mathematically smother [104]–[106] compared to the local theory. That is, for the local theory, entropy condition must be introduced to single out the physically relevant solution; information is lost over time otherwise and weak solutions are thus not unique. Information is not lost over time in the nonlocal theory, and thus weak solutions of nonlocal conservation laws are naturally unique.

Nonlocal conservation laws can be used to describe the dynamics of a large spectrum of physical phenomena where some quantity is conserved while interacting nonlocally. This makes them suitable to model traffic flow in transportation systems for instance [107]–[109], model nonlocal multi-class traffic flow [110], and study the smoothing of the nonlocal term and shock formation [111]. They have also been used to study traffic modelling from nonlocal follow the leader ordinary differential equations [112] and the corresponding traveling waves complemented in [113].

Nonlocal conservation laws have also been used to model and study supply chains [114]–[116]. Generalization to multi-commodity models on networks have been developed in [117], [118], and material flow in conveyor belts have been presented in [119], [120]. They have also been used as a tool in several other domains such as biological and industrial modeling [121], and crowd dynamics [122].

Nonlocal conservation law model

In this chapter, we develop a general numerical solver scheme based on a triangulation of the characteristic for the the class of two-dimensional nonlocal conservation laws presented in Definition 5.2.2. The same theory could be extended to solve higher dimensional models, but we leave outside the scope of this manuscript.

Assumption 5.2.1 (Input datum). *In this work, assume the following:*

- $T \in \mathbb{R}_{>0}$.
- $\Omega \coloneqq \mathbb{R}^2$.
- $\Omega_T \coloneqq (0, T) \times \Omega$.
- $\boldsymbol{q}_0 \in BV(\Omega; \mathbb{R})$.
- $h : \Omega_T \times \Omega$ *is a compactly supported Lipschitz continuous kernel.*

Definition 5.2.2 (The problem considered: multi-dimensional nonlocal conservation laws). *In this work we consider the following* ***PDEs*** *in the variables* $(t, \boldsymbol{x}) \in \Omega_T$ *recalling Assumption 5.2.1, where* $q : \Omega_T \to \mathbb{R}$ *is density:*

$$q_t(t, \boldsymbol{x}) + \operatorname{Div}\Big(\boldsymbol{H}[q](t, \boldsymbol{x}) q(t, \boldsymbol{x})\Big) = 0$$
$$q(0, \boldsymbol{x}) = q_0(\boldsymbol{x})$$

and define the nonlocal operator

$$W[q](t, \boldsymbol{x}) = \int_\Omega h(t, \boldsymbol{x}, \tilde{\boldsymbol{x}}) q(t, \tilde{\boldsymbol{x}}) \mathrm{d}\tilde{\boldsymbol{x}}$$
$$\boldsymbol{H}[q](t, \boldsymbol{x}) = \boldsymbol{V}(W[q](t, \boldsymbol{x}))$$

As we want to have solutions which can be discontinuous we require a weaker form of solutions, so called weak solution. To obtain the definition of weak solution, we multiply the previous PDE with a sufficiently smooth test function and integrate over the entire space-time horizon. After an integration by parts, the derivatives are now on the test functions and we then postulate that the corresponding integral equation holds for all test functions from a sufficiently large set. In equations, take a $\varphi : [0, T] \times \Omega \to \mathbb{R}$ so that $\varphi(T, x) = 0\ \forall \boldsymbol{x} \in \Omega$ and $\lim_{|\boldsymbol{x}| \to \infty} \varphi(t, \boldsymbol{x}) = 0\ \forall t \in [0, T]$ and write:

$$\varphi(t, \boldsymbol{x}) q_t(t, \boldsymbol{x}) + \operatorname{Div}\Big(\boldsymbol{H}[q](t, \boldsymbol{x}) q(t, \boldsymbol{x})\Big) \varphi(t, \boldsymbol{x}) = 0$$

An integration over space-time and then, the integration by parts to shift derivatives to the test-function φ leads to:

$$\iint_{\Omega_T} \varphi(t, \boldsymbol{x}) q_t(t, \boldsymbol{x}) + \operatorname{Div}\Big(\boldsymbol{H}[q](t, \boldsymbol{x}) q(t, \boldsymbol{x})\Big) \varphi(t, \boldsymbol{x})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}t = 0 \tag{5.1}$$

integration by parts leads to what we consider as the weak solution formulation:

$$\begin{aligned} -\iint_{\Omega_T} \varphi_t(t,\boldsymbol{x})q(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \iint_{\Omega} \left[\varphi(t,\boldsymbol{x})q(t,\boldsymbol{x})\right]_{t=0}^{t=T} \mathrm{d}\boldsymbol{x} \\ -\iint_{\Omega_T} \boldsymbol{H}[q](t,\boldsymbol{x})q(t,\boldsymbol{x})\nabla\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t = 0 \end{aligned} \tag{5.2}$$

$$\begin{aligned} \iint_{\Omega_T} \varphi_t(t,\boldsymbol{x})q(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \iint_{\Omega} \varphi(0,\boldsymbol{x})q_0(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\ +\iint_{\Omega_T} \boldsymbol{H}[q](t,\boldsymbol{x})q(t,\boldsymbol{x})\nabla\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t = 0 \end{aligned} \tag{5.3}$$

Theorem 5.2.3 (Well-posedness of the weak solution and the characteristics)**.** *Let Assumption 5.2.1 hold and assume that $q \in C([0,T];L^1(\mathbb{R}))$ is given. For any smooth test function $\varphi : [0,T] \times \Omega \to \mathbb{R}$ so that $\varphi(T,x) = 0\ \forall \boldsymbol{x} \in \Omega$ and $\lim_{|\boldsymbol{x}|\to\infty} \varphi(t,\boldsymbol{x}) = 0\ \forall t \in [0,T]$, the weak solution, defined in Eq.* (5.3)*, is well-posed and admits a unique solution. The following integral equation in $\boldsymbol{\xi} : \Omega_T \times (0,T) \to \Omega$ is the characteristics in integrated form:*

$$\boldsymbol{\xi}(t,\boldsymbol{x};\tau) = \boldsymbol{x} + \int_t^\tau \boldsymbol{H}[q](t,\boldsymbol{\xi}(t,\boldsymbol{x};s))\,\mathrm{d}s \tag{5.4}$$

and the solution can then be defined in terms of this characteristics

$$q(t,\boldsymbol{x}) = q_0(\boldsymbol{\xi}(t,\boldsymbol{x};0))\det(\mathrm{D}_2\boldsymbol{\xi}(t,\boldsymbol{x};0)) \quad \forall(t,\boldsymbol{x}) \in \Omega_T \tag{5.5}$$

$$\int_\Delta q(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_\Delta q_0(\boldsymbol{\xi}(t,\boldsymbol{x};0))\det(\mathrm{D}_2\boldsymbol{\xi}(t,\boldsymbol{x};0))\,\mathrm{d}\boldsymbol{x} = \int_{\boldsymbol{\xi}(0,\Delta;t)} q_0(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{5.6}$$

The proof to this theorem can be found in [105] and in [123].

Motivation for the characteristic solver

The numerical solver presented in this chapter preserve discontinuities compared to classical finite volume solvers, which are inherently numerically dissipative as shown in Figure 5.15 in Section 5.6 (Nondissipative property section). Moreover, the characteristic solver is less sensitive to CFL conditions that are commonly required for classical numerical solvers such as [102]. The numerical solutions using the method presented here maintain structure and the important properties of the solution at different discretization resolutions (mesh resolution and time scales). These properties offer tools to study and understand solutions of nonlocal conservation laws better. Theoretical bases for this approach have been studied in [105], [123]–[125]. To the best of our knowledge, [102] is the only published numerical solver for these models to date of this manuscript.

5.3 General solvers and underlying theory

This section uses Theorem 5.2.3 to develop a general solver for the class of nonlocal conservation laws presented in Definition 5.2.2. The theorem states the existence and uniqueness of solutions to the characteristic curves of these PDE models. These curves represent the basis to solve the PDE models by means of a triangulation evolution in time and a corresponding evolution of respective density profile approximated by this triangulation.

To discritize the domain $\boldsymbol{x}$ of the solution, a meshing scheme is defined next. For a two dimensional domain, this could be represented using a triangulation, which then is evolved in time. This triangulation is then augmented with a density profile evolving in time to represent the solution q. This section starts with general definitions that are later used to derive numerical solver equations.

Definition 5.3.1 (Domain surface triangulation)**.** *Define the triangulation $\mathscr{T}_\Phi$ as the set of triangles $\{\Delta_{(j)} | j \in \{1 \dots J\}\}$ where J is the number of triangles in the triangulation such that $\Delta_{(j)} \subset \mathbb{R}^2$, $\cup_{j=1}^{J} \Delta_{(j)}$ covers some open connected domain $\Phi \subset \mathbb{R}^2$ without holes and, for any $i, j \in \{1 \dots J\}$ and $i \neq j$, $\Delta_{(i)} \cap \Delta_{(j)}$ is empty. Each triangle $\Delta_{(j)}$ is defined by the ordered set of three vertices (or points) $\boldsymbol{p}_{(j,i)}$ where $j \in \{1 \dots J\}$, $i \in \{1 \dots 3\}$ and $\boldsymbol{p}_{(j,i)} \in \mathbb{R}^2$. Let $\boldsymbol{P} \in M \times \mathbb{R}^2$ be the set of unique vertices in the triangulation where M is the number of vertices in this set.*

Definition 5.3.2 (Time evolving triangulation)**.** *Define the time evolving triangulation $\mathscr{T}_\Phi(t)$ as a diffeomorphic manifold approximated by the triangulation $\mathscr{T}_\Phi$ such that the triangles are allowed to move and deform as a function of time t according to some differentiable map operating on traingle vertices. Hence define the evolving triangle $\Delta_{(j)}(t)$ as the triangle represented by the ordered set of three vertices (or points) $\boldsymbol{p}_{(j,i)}(t)$ where $j \in \{1 \dots J\}$, $i \in \{1 \dots 3\}$ and $\boldsymbol{p}_{(j,i)}(t) \in \mathbb{R}^2$. Let $\boldsymbol{P} \in M \times \mathbb{R}^2$ be the set of unique vertices in the triangulation where M is the number of vertices in this set.*

Remark 5.3.3 (Generalization to arbitrary meshing schemes)**.** *We assume triangular meshes in this work for convenience, but they can be generalized to meshes with arbitrary face geometry.*

Definition 5.3.4 (Density profile triangulation)**.** *Define density profile triangulation as the triangulation $\mathscr{T}_\Omega(t)$ and assign some density value $q_{(j)}(t)$ to each $\Delta_{(j)}(t) \in \mathscr{T}_\Omega(t)$ where $j \in \{1 \dots J\}$.*

Remark 5.3.5 (Density profile approximation)**.** *The density profile is approximated as a piece-wise constant surface in Definition 5.3.4. This allows for a simple and efficient numerical solver. Density surface could alternatively be estimated using higher order approximations such as multidimensional trapezoidal rule for higher accuracy.*

These definitions along with Theorem 5.2.3 are used next to derive a discritized solution as follows:

Theorem 5.3.6 (Density solution along characteristic faces)**.** *Define a characteristic face as the region enclosed initially by any given triangle* Δ*. A characteristic face evolution in time is defined as the evolution of each point* $\boldsymbol{x} \in \Delta$ *according to the characteristic curves. Then, along these characteristic faces,* $q(t, \boldsymbol{x})$ *is conserved, and hence:*

$$\int_{\Delta} q(t, \boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \int_{\boldsymbol{\xi}(0,\Delta;t)} q(0, \boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{5.7}$$

Use Definition 5.3.4 to approximate $q(t, \boldsymbol{x})$*, and let each triangle evolve according to the characteristic curves defined in Eq.* (5.4)*, then, for each* $j \in \{1 \dots J\}$*:*

$$q_j(t) \approx q_j(0) \frac{\iint_{\Delta_{(j)}(0)} d\boldsymbol{x}}{\iint_{\Delta_{(j)}(t)} d\boldsymbol{x}} \tag{5.8}$$

For simplicity, we assume that a triangular characteristic face remain a triangle as it evolves according to the characteristic curves. With this assumption, computation of the evolution of the triangles in time simplifies to evolution of the three vertices representing the triangle. For each vertex, this evolution in time could be discritized as follows:

Theorem 5.3.7 (Explicit Euler integration of characteristic curves and faces in time)**.** *Using first order Euler integration, for* $\tau = t + \delta_t$ *and some small discrete step* δ_t*, we discretize the characteristic Eq.* (5.4) *of the PDE in Definition 5.2.2 as follows:*

$$\xi(t, \boldsymbol{x}; \tau) \approx \boldsymbol{x} + \boldsymbol{H}[q(t)](t, \boldsymbol{x})\delta_t \tag{5.9}$$

Remark 5.3.8 (Time discritization of the characteristic curves)**.** *Higher order integration schemes such as Runge–Kutta could be used as well. However, in this chapter, we use Euler integration for simplicity and low computational complexity.*

Computation of the characteristic curves in Eq. (5.9) require a computation of the nonlocal convolution. By linearity of the integral operator in W defined in Definition 5.2.2, we could evaluate the convolution for each triangle separately. The computation of the convolution could be simplified by utilizing the assumption that q is constant inside each triangle. If explicit computation of convolution of the nonlocal term h is tedious, one could further simplify the computation by assuming that h is constant inside each triangle. This would lead to the following:

Theorem 5.3.9 (Discritization of the nonlocal convolution)**.** *Consider the nonlocal convolution defined in Definition 5.2.2:*

$$W[q](t, \boldsymbol{x}) = \iint_{\Omega} h(t, \boldsymbol{x}, \tilde{\boldsymbol{x}}) q(t, \tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \tag{5.10}$$

Let $\mathscr{T}_\Omega(t)$ be a triangulation as defined in Definition 5.3.2 representing the integral domain Ω, then the nonlocal convolution could be written down as:

$$W[q](t, \boldsymbol{x}) = \sum_{j=0}^{J} \iint_{\Delta_{(j)}(t)} h(t, \boldsymbol{x}, \tilde{\boldsymbol{x}}) q(t, \tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \tag{5.11}$$

Assuming q is constant inside each triangle, the convolution simplifies to:

$$\sum_{j=0}^{J} q_{(j)}(t) \iint_{\Delta_{(j)}(t)} h(t, \boldsymbol{x}, \tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \tag{5.12}$$

where the double integral could be computed explicitly or reduced to a single integral using divergence theorem. Otherwise, the convolution could be further simplified by taking the two-dimensional Riemann sum approximation to get:

$$\sum_{j=0}^{J} q_{(j)}(t) h_{(j)}(t, \boldsymbol{x}) \iint_{\Delta_{(j)}(t)} d\tilde{\boldsymbol{x}} \tag{5.13}$$

Where $h_{(j)}(t, \boldsymbol{x})$ is some discretization scheme of the nonlocal kernel (such as the average value inside triangle j or the value of the kernel at the centroid of the triangle), and $\iint_{\Delta_{(j)}(t)} d\tilde{\boldsymbol{x}}$ represent the area of each respective triangle, which can be computed explicitly in a trivial way.

Discontinuities in Definition 5.2.2 are introduced with the initial condition but are neither created or destroyed as the solution evolves in time; as can be observed from Theorem 5.2.3 where the solution represents transportation of the initial datum multiplied by smooth functions. Accurate representation of these discontinuities in the solution rely on properly approximating these discontinuities into the triangulation defined in Definition 5.3.1 and the solver presented in this section inherently maintains their evolution in time. This could be enforced by extending the definition of the triangulation into a triangulation constrained by the discontinuity boundary as shown in Figure 5.4 and Figure 5.5.

The evolution of the mesh in time causes deformation, and because of discretization, this potentially leads to degeneracy or degradation of mesh quality. To maintain the quality of the solution, a CFL type condition to track mesh quality is necessary. This condition would evaluate mesh properties such as triangle overlapping, triangle malformation, etc. Triangles overlap happen when the order of vertices in the triangulation is violated while evolving them in time as shown in Figure 5.9. This can occur when the time step δ_t of the explicit Euler is chosen inappropriately too large. Triangle malformation stems from model dynamics that either concentrate density beyond the handling of numerical tools as shown in Figure 5.8 or spread it until density cannot be precisely represented without refinement. When the mesh degrades, the mesh must be restored using a process that conserves the L^1 mass of the solution and preserves the discontinuity boundary.

The solver presented in this section is L^1 conserving that does not destroy discontinuities. It approximates the solution to the initial value problem of the PDE model by means of a moving in time mesh, a property that could support efficiency; we only need to triangulate and solve for the non-zero regions of the domain. Moreover, the solution is constructed by a naturally mesh adaptive way; the mesh deforms according to the dynamics.

The solver in this section introduces several types of error outlined as follows:

Error 1: Triangulation of initial datum: this is introduced due to the discritization of the discontinuity boundary and the discritization of the value of q. For example, a square boundary can be approximated accurately, however, a circle can only be represented approximately with the coordinates used in this section. We also assume that the value of q is piecewise constant within each triangle.

Error 2: Computation of the nonlocal term: this section suggested approximation of the nonlocal kernel h as a piecewise constant within each triangle, which could introduce error into the nonlocal operator $\boldsymbol{H}$. This could be remedied by explicit integration instead.

Error 3: Geometric approximation of characteristic face evolution in time: this section assumes that a triangle in $\boldsymbol{x}$ at some time instance t remains a triangle as it evolve according to the characteristic equation in time. This assumption will introduce error into the solution. However, for characteristic equations with small lipschitz-constant, and for sufficiently small time steps, deformation triangle geometry and hence the associated error is negligible.

Error 4: Mesh maintenance: error could be introduced for approximate restorations of degenerate meshes. This could be remedied by increase in algorithm and computational complexity.

5.4 Vectorized representation

This section presents the fundamental equations necessary to implement the solver in a vectorized form. The discrete time characteristic equation in Eq. (5.9) is cast as the triangulation vertex update equation, for each point $\boldsymbol{p}_{(i)}^{(k)} \in \boldsymbol{P}^{(k)}$ where $i \in 1 \dots M$ and k is the discrete time step, as follows:

$$\boldsymbol{p}_{(i)}^{(k+1)} \approx \boldsymbol{p}_{(i)}^{(k)} + \boldsymbol{H}[q^{(k)}](\boldsymbol{p}_{(i)}^{(k)})\delta_t \tag{5.14}$$

Hence, the update equation for all vertices of the triangulation can be represented in vector form as follows:

$$\boldsymbol{P}^{(k+1)} \approx \boldsymbol{P}^{(k)} + \boldsymbol{H}[q^{(k)}](\boldsymbol{P}^{(k)})\delta_t \tag{5.15}$$

where $\boldsymbol{P}$ is an $M \times 2$ where each row is a vertex from the triangulation represented by its $2D$ coordinates (x, v) as follows:

$$\boldsymbol{P}^{(k)} = \begin{bmatrix} \boldsymbol{p}^{(k)}_{(1)} \\ \boldsymbol{p}^{(k)}_{(2)} \\ \vdots \\ \boldsymbol{p}^{(k)}_{(M)} \end{bmatrix} = \begin{bmatrix} x^{(k)}_{(1)} & v^{(k)}_{(1)} \\ x^{(k)}_{(2)} & v^{(k)}_{(2)} \\ \vdots \\ x^{(k)}_{(M)} & v^{(k)}_{(M)} \end{bmatrix} \tag{5.16}$$

The $\boldsymbol{H}$ is an $M \times 2$ matrix representing the nonlocal operator evaluated at each respective vertex of the triangulation; noting that $\boldsymbol{H}[q^{(k)}](\boldsymbol{P}^{(k)})$ represent element-wise evaluation of $\boldsymbol{H}$ for each vertex in $\boldsymbol{P}^{(k)}$. This operator is evaluated given a user defined function $\boldsymbol{V}(W)$. However, in general, for each vertex, the nonlocal convolution W could be evaluated using Theorem 5.3.9 in following vector form:

$$W^{(k)}[q](\boldsymbol{x}) = \sum_{j=0}^{J} q^{(k)}_{(j)} \tilde{h}^{(k)}_{(j)}(\boldsymbol{x}) \tag{5.17}$$

$$= \begin{bmatrix} \tilde{h}^{(k)}_{(1)}(\boldsymbol{x}) & \tilde{h}^{(k)}_{(2)}(\boldsymbol{x}) & \cdots & \tilde{h}^{(k)}_{(J)}(\boldsymbol{x}) \end{bmatrix} \cdot \begin{bmatrix} q^{(k)}_{(1)} \\ q^{(k)}_{(2)} \\ \vdots \\ q^{(k)}_{(J)} \end{bmatrix} \tag{5.18}$$

where

$$\tilde{h}^{(k)}_{(j)}(\boldsymbol{x}) = \iint_{\Delta^{(k)}_{(j)}} h^{(k)}(\boldsymbol{x}, \tilde{\boldsymbol{x}})\, \mathrm{d}\tilde{\boldsymbol{x}} \tag{5.19}$$

$$\approx h^{(k)}_{(j)}(\boldsymbol{x}) \iint_{\Delta^{(k)}_{(j)}} \mathrm{d}\tilde{\boldsymbol{x}} \tag{5.20}$$

The discretization of $h^{(k)}_{(j)}$ is chosen as appropriate for the problem. For instance, this could be the average value inside triangle j or the value of the function evaluated at the centroid of the triangle.

This $W^{(k)}[q](\boldsymbol{x})$ will have to be evaluated for each $\boldsymbol{x} \in \boldsymbol{P}^{(k)}$ hence, in vector form:

$$W^{(k)}[q](\boldsymbol{P}^{(k)}) = \begin{bmatrix} W^{(k)}[q](\boldsymbol{p}^{(k)}_{(1)}) \\ W^{(k)}[q](\boldsymbol{p}^{(k)}_{(2)}) \\ \vdots \\ W^{(k)}[q](\boldsymbol{p}^{(k)}_{(M)}) \end{bmatrix} = \begin{bmatrix} \hat{h}^{(k)}_{(1,1)} & \hat{h}^{(k)}_{(1,2)} & \cdots & \hat{h}^{(k)}_{(1,J)} \\ \hat{h}^{(k)}_{(2,1)} & \hat{h}^{(k)}_{(2,2)} & \cdots & \hat{h}^{(k)}_{(2,J)} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{h}^{(k)}_{(M,1)} & \hat{h}^{(k)}_{(M,2)} & \cdots & \hat{h}^{(k)}_{(M,J)} \end{bmatrix} \cdot \begin{bmatrix} q^{(k)}_{(1)} \\ q^{(k)}_{(2)} \\ \vdots \\ q^{(k)}_{(J)} \end{bmatrix} \tag{5.21}$$

where the matrix $\left[\hat{h}^{(k)}_{(i,j)}\right]$ is the matrix representing $\tilde{h}^{(k)}_{(j)}$ evaluated for vertex i and integrated over triangle j as follows:

$$\hat{h}^{(k)}_{(i,j)} = \tilde{h}^{(k)}_{(j)}(\boldsymbol{p}^{(k)}_i) \tag{5.22}$$

This matrix can be generated with a properly formatted argument matrices and using element-wise operations.

Density of each triangle can be updated using the equation from Theorem 5.3.6 as follows:

$$q^{(k+1)}_j \approx q^{(k)}_j \frac{\text{area}\left(\Delta^{(k)}_{(j)}\right)}{\text{area}\left(\Delta^{(k+1)}_{(j)}\right)} \tag{5.23}$$

where the area of a triangle is half the absolute value of its determinant, which can be evaluated given the coordinates of its vertices $(x_i, v_i) \in \mathbb{R}^2$ for $i \in \{1 \ldots 3\}$ as follows:

$$\frac{|x_1(v_2 - v_3) + x_2(v_3 - v_1) + x_3(v_1 - v_2)|}{2} \tag{5.24}$$

Density scaling equation can be vectorized using element wise operations or using matrix operations for properly formatted matrices.

5.5 Solver algorithm

This section presents an outline for the characteristic-based solver algorithm for Definition 5.2.2 utilizing the theory presented in the earlier sections. The core of the algorithm is an iterative density profile mesh evolving process presented in Algorithm 2. This core solver takes an initial density profile mesh $\{P, ConList, Density\}$, the desired time horizon T, and initial time step size δ_t as an input, and generates the solution q as an output. In addition to the primary solver equations to evolve the mesh, time step adaptation and mesh maintenance are performed to correct for mesh degeneracy.

This section also presents support algorithms to the core solver; namely, Algorithm 1, an algorithm to generate discontinuity preserving initial density meshes to use as an input for Algorithm 2, and Algorithm 3, an algorithm to restore the density profile mesh to use to correct for mesh degeneracy.

Algorithm 1 takes, as input, desired domain boundary approximated as a triangle and defined by two corner points; triangulation resolution approximated by the desired number of unique triangle vertices in the mesh; an initial density function that is queryable given any point within the domain; mesh face density estimation function such as a function that evaluates the density function at the centroid of the provided triangle; and discontinuity boundary for discontinuous initial density functions approximated by a set of points and a set of edges edges. This algorithm uses domain boundary and resolution to generate vertices, which are used to generate a Delaunay Triangulation [126], or Constrained Delaunay

Triangulation [127] constrained by discontinuity boundary for discontinuous initial density functions. Note that the initial density value of each triangle is presented in the algorithm outline with a for loop. This for loop could be vectorized for simple estimation function such as those evaluating density at triangle centroids. This function outputs E_{disc} to represent and keep track of discontinuity boundary. This set references points from P and hence remain a static set during integration and can be used to extract P_{disc} after they have had evolved in time when needed.

The solver algorithm presented in this section causes mesh deformation with potential for degeneracy as discussed earlier in this chapter. Triangle overlapping could be corrected for by adapting the time step, and triangle malformation could be corrected for using an algorithm such as Algorithm 3. Note that this mesh maintenance algorithm is not L^1 mass conserving. To conserve L^1 mass after remeshing, $Q_{\text{est}}[Q_o](\text{face}_i)$ function passed from Algorithm 3 to Algorithm 1 must be designed carefully for this purpose. Otherwise, the remeshed surface could be approximated using a simple piece-wise constant interpolation function $Q_{\text{est}}[Q_o](\text{face}_i)$ as already assumed for initial density profile approximation.

Algorithm 1: Simple discontinuity preserving initial density mesh generation algorithm

```
Input  : Initial domain boundary $\{x_{1,\min}, x_{1,\max}, x_{2,\min}, x_{2,\max}\}$
Input  : Triangulation resolution—number of mesh vertices along each
         dimension—$\{x_{1,\text{resolution}}, x_{2,\text{resolution}}\}$
Input  : Queryable initial density function $Q_o(\boldsymbol{x})$
Input  : Mesh face density estimation function $Q_{\text{est}}[Q_o](\text{face}_i)$
Generate sample points $P_{\text{mesh}}$ inside initial domain at defined resolution;
if $Q_o$ is discontinuous then
    Input  : Discontinuity boundary set defined as set of points $P_{\text{disc}}$ and an
             ordered set of edges $E_{\text{disc}}$ from the point set $P_{\text{disc}}$
    Generate $P$ and ConList using Constrained Delaunay Triangulation of the
      augmented point set $\{P_{\text{mesh}} \cup P_{\text{disc}}\}$ constrained by the discontinuity boundary
      set $\{P_{\text{disc}}, E_{\text{disc}}\}$;
else
    Generate $P$ and ConList using Delaunay Triangulation of the point set $P_{\text{mesh}}$;
    Let $E_{\text{disc}}$ be an empty set;
end
for ( $j = 0$; $j \leq |ConList|$; $i = i + 1$ ) {
    Generate initial density profile $Density_j = Q_{\text{est}}[Q_o](\text{face}_j)$;
}
Output: Density profile mesh $\{P, ConList, Density, E_{\text{disc}}\}$
```

Algorithm 2: Core solver for PDE Model in Definition 5.2.2

```
Input  : Initial density profile mesh $\{P, ConList, Density, E_{\text{disc}}\}$
Input  : Time horizon $T$ and time step $\Delta_t$
for ( $t = 0; t \le T; t = t + \Delta_t$ ) {
    Move triangles: integrate characteristic curves using Eq. (5.15);
    Solve for densities Eq. (5.23);
    if triangle overlap then
        Adapt time step $\Delta_t \leftarrow \frac{\Delta_t}{2}$;
        Undo step;
    end
    if triangle malformation then
        Perform mesh maintenance;
        Undo step;
    end
}
Output: PDE solution $q$
```

Algorithm 3: Discontinuity preserving mesh maintenance

```
Input  : Degenerate density profile mesh $q$ as $\{P, ConList, Density, E_{\text{disc}}\}$
if $E_{disc}$ is not empty then
    Extract discontinuity boundary $P_{\text{disc}}$ from $P$ given $E_{\text{disc}}$;
end
Apply Algorithm 1 to get $q_{\text{remeshed}}$ by providing $q$ as $Q_o$, some interpolation function
  $Q_{\text{est}}$, and other inputs as appropriate;
Output: Remeshed density profile $q_{\text{remeshed}}$
```

5.6 Numerical implementation

This section presents numerical examples for Definition 5.2.2. We first validate the solver by comparing its results against an explicit solution. We then use the algorithm to develop numerical solutions to a traffic flow model and to Lagrangian traffic actuation. Finally, we compare the results to a classical solver based on viscosity approximation to illustrate the nondissipative property of our algorithm.

Validation against an explicit solution

This section validates the solver against an explicit solution to the model in Example 5.6.1 solved in [105]. In this model, the nonlocal operator computes the overlap of q with a fixed in $\boldsymbol{x}$ integral region with a quadratic support. With a square-shaped initial datum $q(0, \boldsymbol{x})$, the initial datum moves diagonally in $\boldsymbol{x}$ from one corner of the nonlocal integral region to stop at the other corner with a variable speed that peaks while the mass of q is fully inside the nonlocal integral region.

Example 5.6.1 (Moving solution with quadratic support)**.** *Consider the following model in the variables $(t, \boldsymbol{x}) \in [0, T] \times \mathbb{R}^2$ for a sufficiently large T:*

$$\begin{aligned} q_t(t, \boldsymbol{x}) + \operatorname{Div}\Big(\boldsymbol{H}[q](t, \boldsymbol{x}) q(t, \boldsymbol{x})\Big) &= 0 \\ q(0, \boldsymbol{x}) &= \chi_{[-\frac{1}{2}, \frac{1}{2}]^2} \end{aligned}$$

and the following nonlocal operator:

$$\begin{aligned} W[q](t, \boldsymbol{x}) &= \int_0^2 \int_0^2 q(t, \boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ \boldsymbol{H}[q](t, \boldsymbol{x}) &= W[q](t, \boldsymbol{x}) \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

The solution of the equation can be stated as:

$$q(t, \boldsymbol{x}) = \chi_{[\xi(t) - \frac{1}{2}, \xi(t) + \frac{1}{2}]^2}(\boldsymbol{x})$$

where χ is the indicator function and:

$$\xi(t) := \begin{cases} \dfrac{1}{2-t} - \dfrac{1}{2}, & t \in [0, 1] \\ t - \dfrac{1}{2}, & t \in (1, 2] \\ 2.5 - \dfrac{1}{t-1}, & t \in [2, T] \end{cases}$$

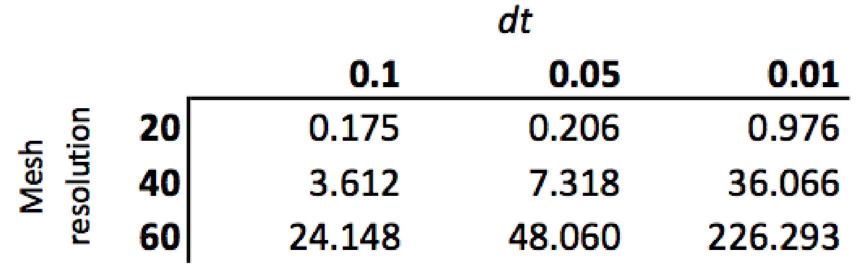
The explicit and numerical solution is shown in Figure 5.1. In this figure, the numerical solution is evaluated for $dt = 0.01$ and a mesh resolution of 40 samples per dimension as outlined in the algorithm section. The numerical error in this example is primarily due to the approximation of the nonlocal convolution. The error could be reduced by computing an explicit solution to the nonlocal term, or by increasing mesh resolution, which could be observed in Figure 5.2.

In Figure 5.2, evaluate L^2 mass of the error between the explicit solution and the numerical solution for a several combinations choices of dt and mesh resolutions. The correlation between mesh resolution and solution accuracy can be observed.

It can be noted that the L^2 mass of the error tend to drop towards the end of the simulation, especially for high resolution meshes. Error in this model is primarily caused by estimation of the nonlocal operator while the structure and discontinuities of the initial datum are preserved. The effect of this error diminish towards the end of the simulation due to the dynamics of the model. While this decay of the L^2 mass of the error cannot be observed for classical solvers, which smooth the initial datum over time, this is not a generally expected behavior for this solver for all models. Hence more complex models must be studied to validate the solver.

Computational time is shown in Table 5.1. The computational time is estimated for solving the model with $T = 4$ and with dt and mesh resolutions as shown in the table. The initial mesh covers covers $[-0.6, 0.6]^2$ domain. The nonlocal convolution is the most compute extensive step to solve this model, which scales quadratically with the resolution of the two-dimensional mesh as will be shown in more details in the following sections.

Table 5.1: Computational time in seconds to numerically solve the model in Example 5.6.1.



		dt		
		0.1	**0.05**	**0.01**
Mesh resolution	**20**	0.175	0.206	0.976
	40	3.612	7.318	36.066
	60	24.148	48.060	226.293

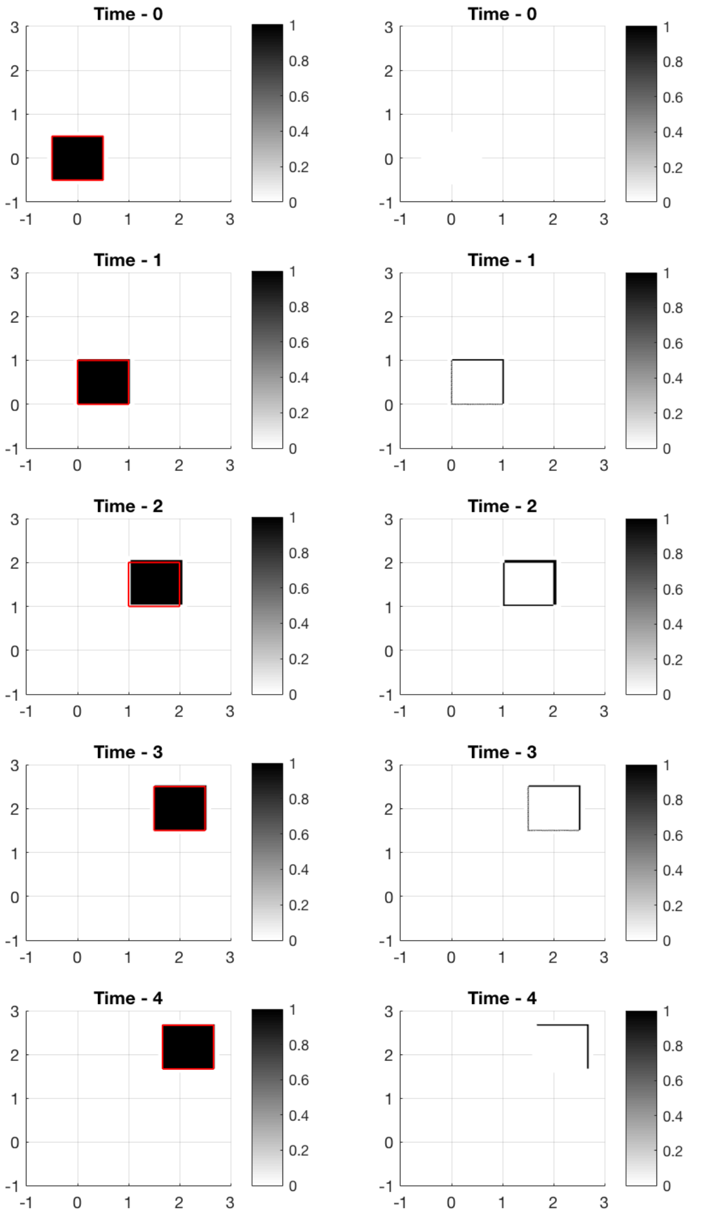
Traffic flow model

In this section, we use the algorithm developed in this chapter to solve the PDE model of traffic flow presented in Example 5.6.2. This model is a mean-field limit adapted from [99] to describe mesoscopic freeway traffic flow dynamics using the microscopic intelligent driver model (IDM) [128]. In this model, q encodes the density of vehicles driving at a given speed and located at a given spatial position along the modeled freeway.

Example 5.6.2 (mesoscopic traffic flow model)**.**

$$q_t(t,x,v) + v\partial_x q(t,x,v) = -\partial_v\left(\big(\mathcal{H}[q(t)](x,v)\big)q(t,x,v)\right) \tag{5.25}$$

$$q(0,x,v) = q_0(x,v), \quad (x,v) \in \mathbb{R} \times \mathbb{R}_{>0} \tag{5.26}$$



(a) (b)

Figure 5.1: Comparison of numerical solution to an explicit solution to Example 5.6.1. Evolution of solution in time is shown in (a) where the filled square is the numerical solution and the red outline is the explicit solution. The evolution of absolute error in time is shown in (b).

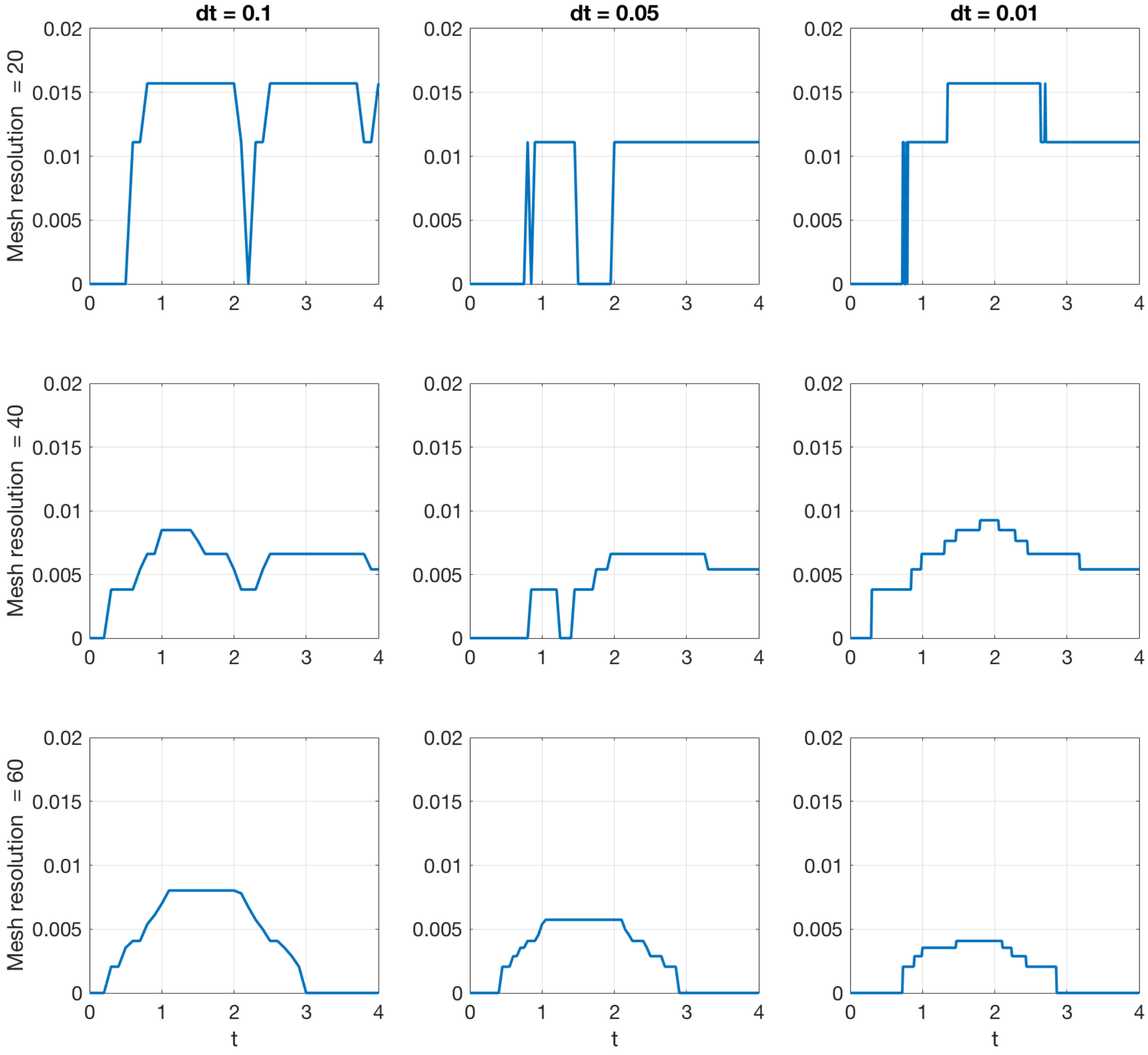


Figure 5.2: Evolution of L^2 mass of error between numerical solution and explicit solution as a function of time for Example 5.6.1.

with the nonlocal operators a defined in

$$\begin{aligned}\mathcal{H}[q](x,v) &= a - a\left(\tfrac{v}{v_0}\right)^{\delta} \\ &\quad - a\int_{\mathbb{R}}\int_{\mathbb{R}_{>0}} h(x-\tilde{x})q(\tilde{x},\tilde{v})\left(\frac{s_0 + vT + \frac{v(v-\tilde{v})}{\sqrt{2ab}}}{\tilde{x}-x-l}\right)^2 \mathrm{d}\tilde{v}\,\mathrm{d}\tilde{x}. \end{aligned} \tag{5.27}$$

$$h(x) := \begin{cases} \mathrm{e}^{-\frac{1}{\left(\frac{\varepsilon_0}{2}\right)^2 - \left(-x-\frac{\varepsilon_0}{2}\right)^2}}, & \textit{if } -\varepsilon_0 < x < 0, \\ 0, & \textit{otherwise.} \end{cases} \tag{5.28}$$

$$\tag{5.29}$$

where v_o, δ, s_o, T, a, b, l, and ε_o are tuning parameters.

Numerical solutions to this model for a rectangular initial datum are shown in Figure 5.3 for mesh resolutions 20 and 40 and time step $dt \in \{0.1, 0.01\}$. This example represent the flow of a stream of vehicles. Vehicles move in space at their respective speeds and accelerate according to the IDM model. In Figure 5.4, we add a second rectangular region into the initial datum. This initial datum represents a fast stream of vehicles approaching a slower one. It can be observed from the solution that the faster stream slows down as it approach the slower stream, while the slower stream speeds up according to the IDM model.

In these figures, accuracy is affected by the rectangular discontinuity boundary constraints imposed into low resolution meshes. Figure 5.5 solve the same model and initial datum presented in Figure 5.4 without imposing an explicit discontinuity constraint into the mesh. While the general behavior is still consistent, significant drop in the resolution of the solution can be observed. Overall, solution robustness to mesh resolution and time step can be noted in all figures presented.

The numerical solution is generated by means of an evolving in time mesh as shown in Figure 5.8. In Figure 5.8, given the dynamics over time, density concentrates in $\boldsymbol{x}$ where triangle sizes approach zero causing numerical challenges. With too large time steps, triangle overlap occurs as shown in Figure 5.9.

Computational time to solve this model is shown in Table 5.2. The system is solved for time horizon of one with an initial datum covering $[0, 20]^2$ domain. For mesh resolutions 10, 20, 30, 40, 50, and 60, the number of triangles and unique vertices for each are as follows: (190, 116), (766, 424), (1753, 938), (3129, 1646), (4919, 2562), and (7095, 3670). Computational complexity scales approximately linearly with number of simulated frames ($\frac{1}{dt}$) and quadratically in the number of triangles in the mesh as shown in Figure 5.6 and Figure 5.7. Computational time for meshes of resolution 10 are dominated by software overhead.

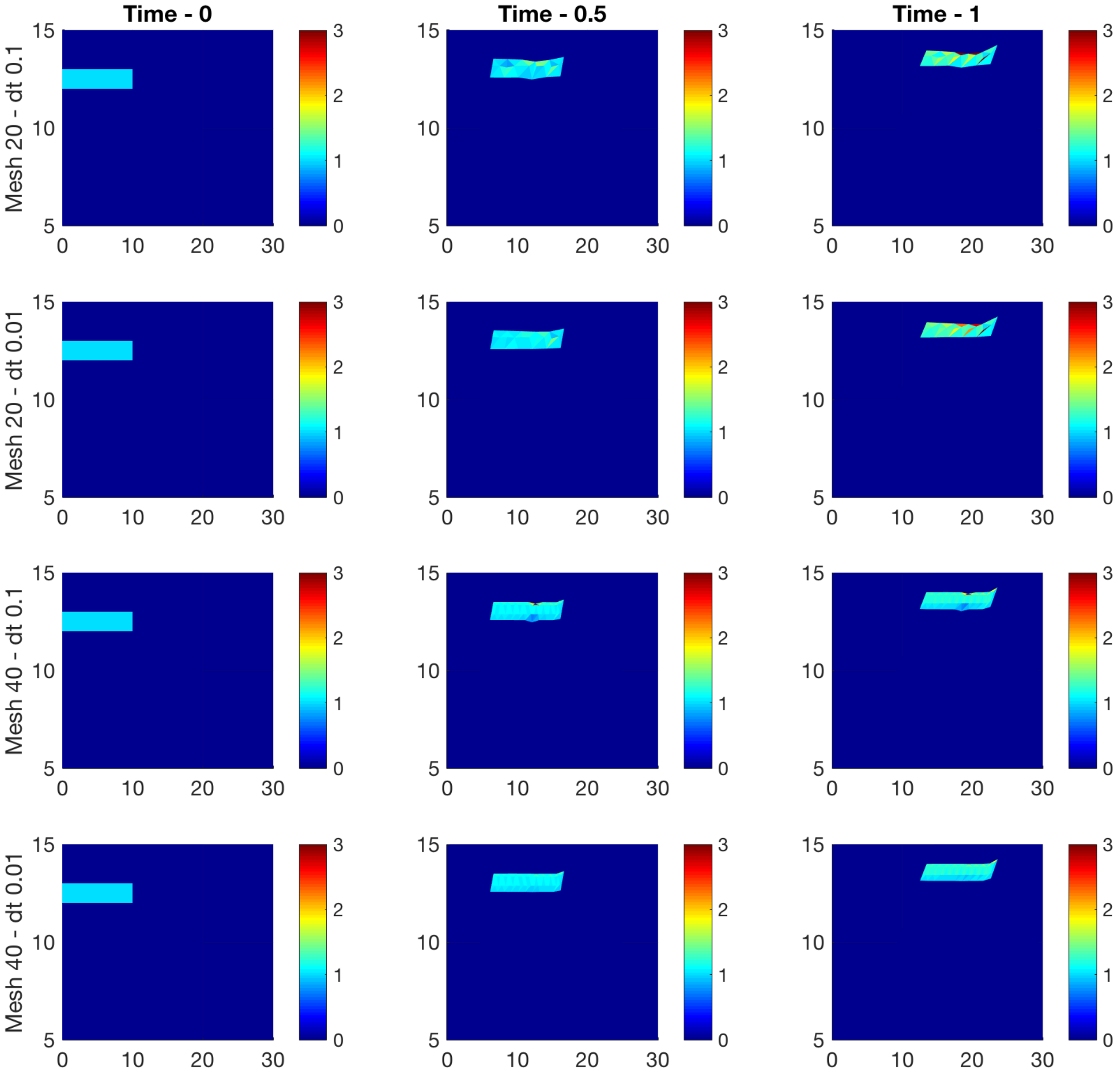


Figure 5.3: Numerical solution to model presented in Example 5.6.2 with a rectangular initial datum.

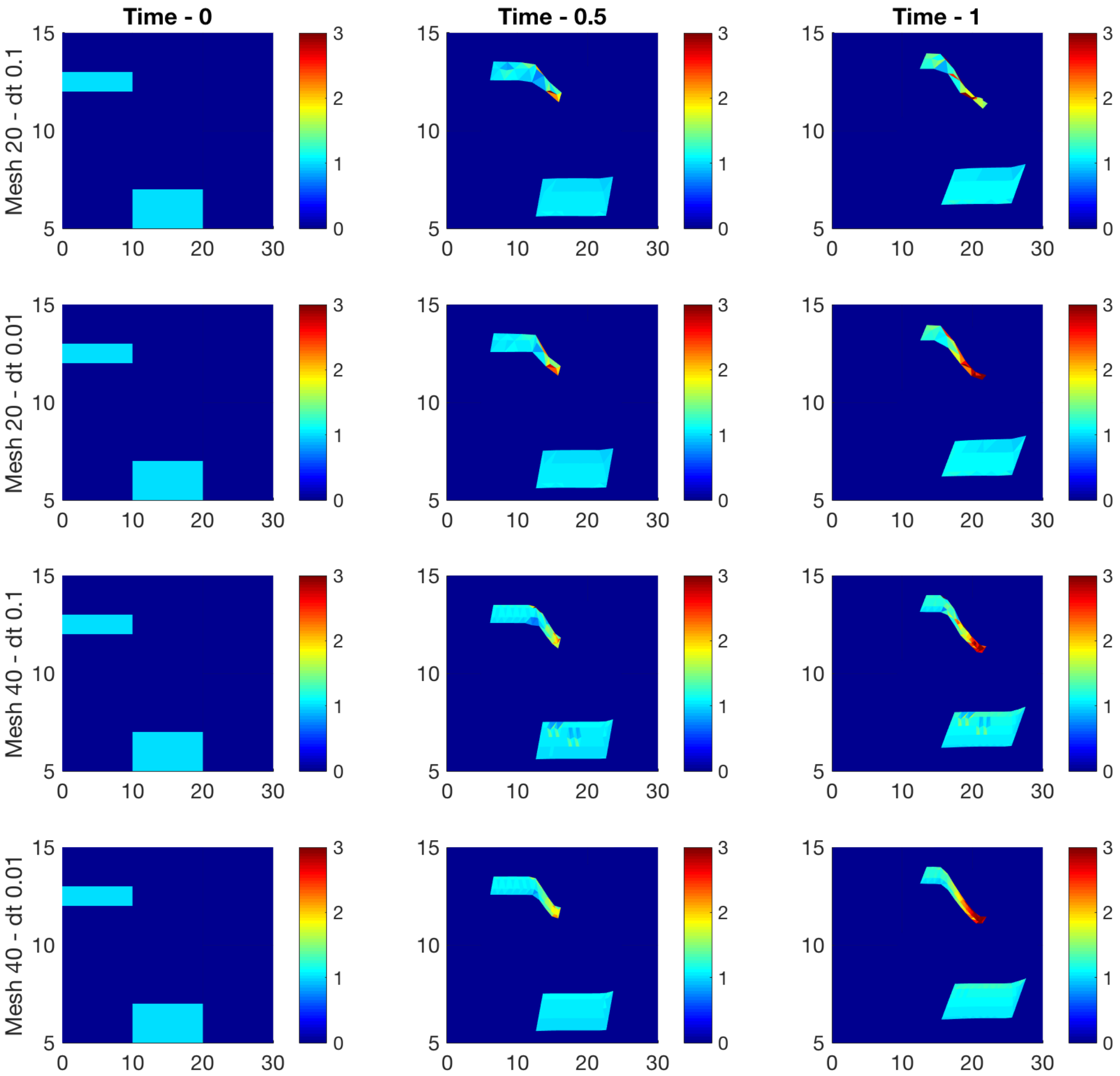


Figure 5.4: Numerical solution to model presented in Example 5.6.2 with two interacting rectangular regions in the initial datum.

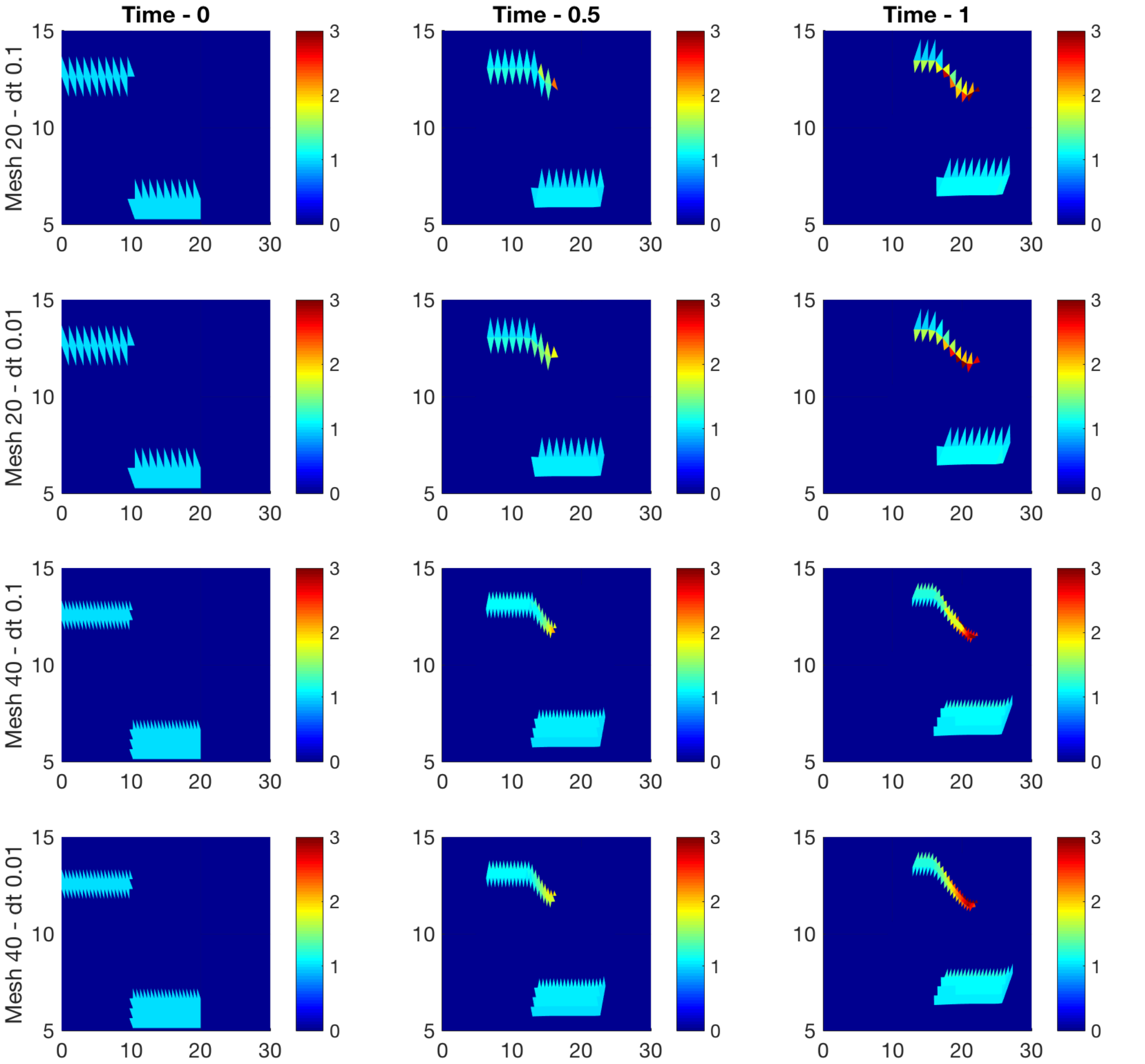
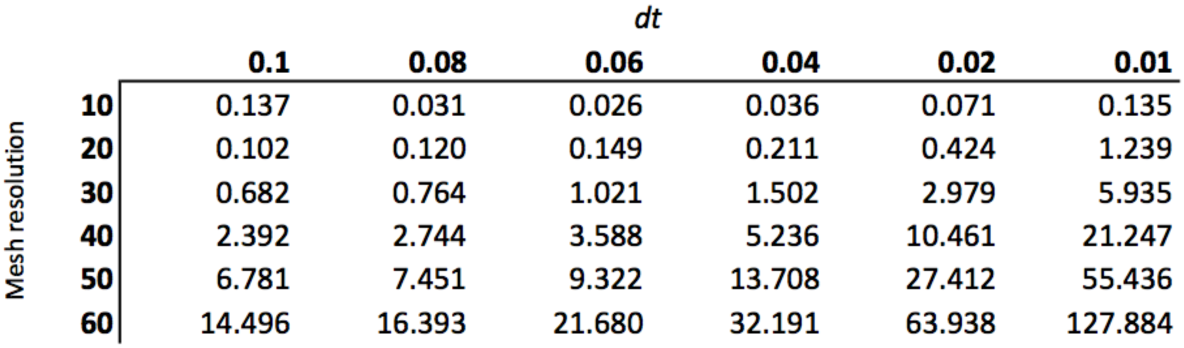


Figure 5.5: Numerical solution to model presented in Example 5.6.2 with two interacting rectangular regions in the initial datum provided to the solver without an explicit discontinuity boundary.

Table 5.2: Computational time in seconds to numerically solve the model in Example 5.6.2.



		dt					
		0.1	**0.08**	**0.06**	**0.04**	**0.02**	**0.01**
Mesh resolution	**10**	0.137	0.031	0.026	0.036	0.071	0.135
	20	0.102	0.120	0.149	0.211	0.424	1.239
	30	0.682	0.764	1.021	1.502	2.979	5.935
	40	2.392	2.744	3.588	5.236	10.461	21.247
	50	6.781	7.451	9.322	13.708	27.412	55.436
	60	14.496	16.393	21.680	32.191	63.938	127.884

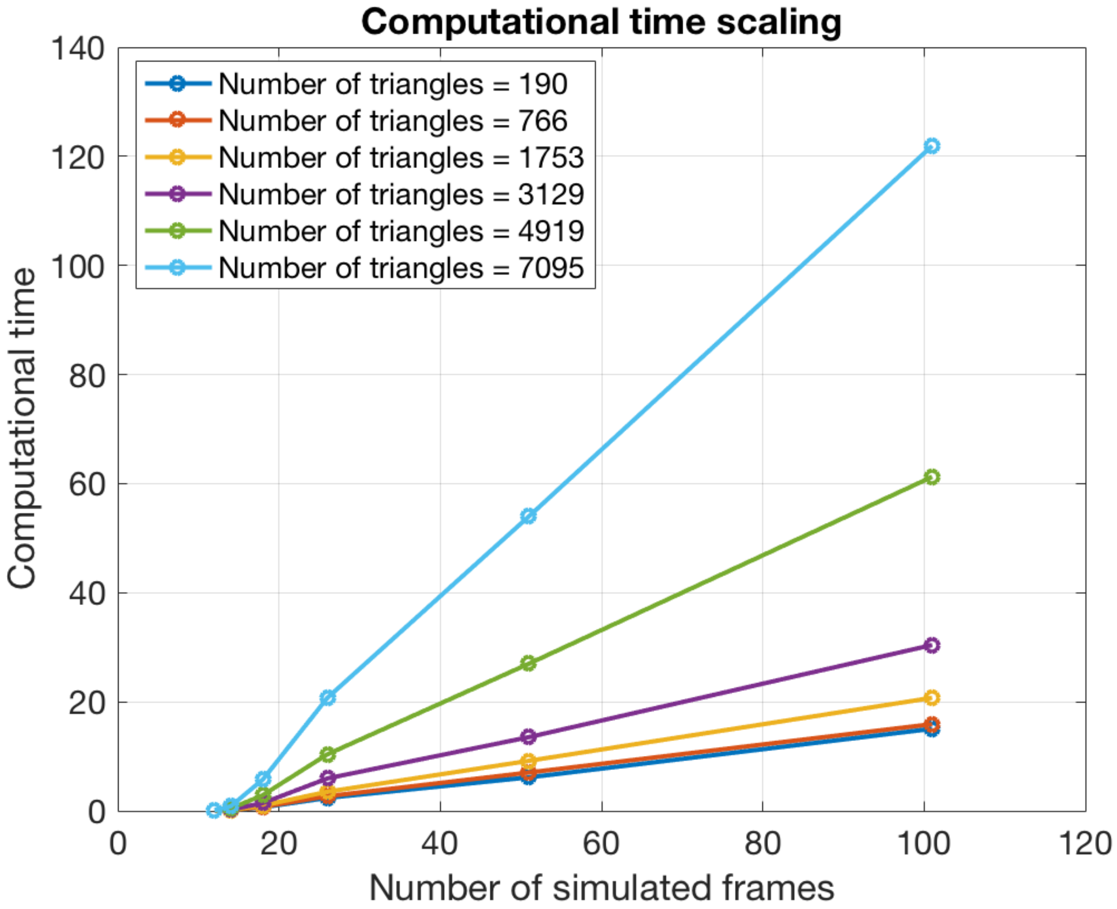


Figure 5.6: Computational time scaling as a function of the number simulated time frames for several mesh resolutions.

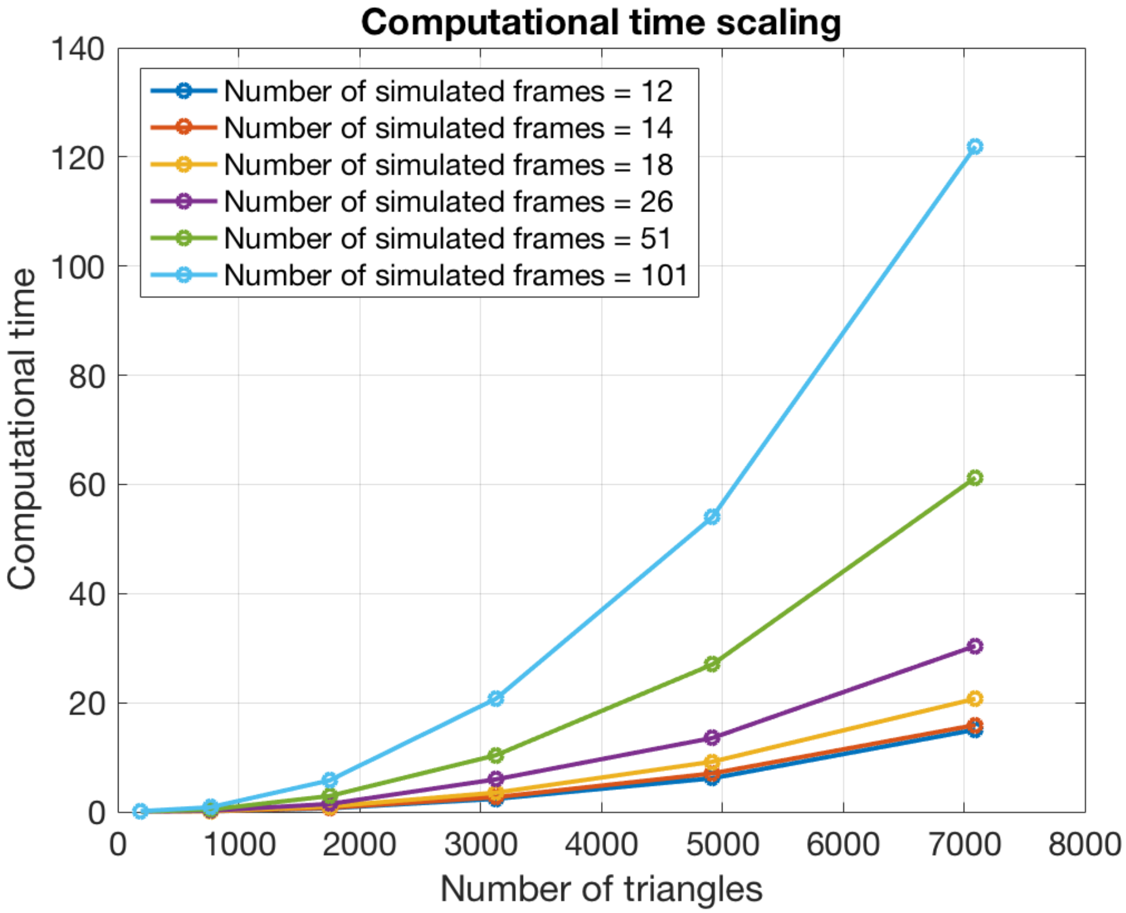


Figure 5.7: Computational time scaling as a function of mesh resolution for several time step settings.

Multi-scale Lagrangian control of traffic flow

In this section, we extend the solver algorithm presented in this chapter in order to solve the coupled ODE-PDE mean-filed system model presented in Example 5.6.3. This model extends the model presented in Example 5.6.2 to describe multi-scale dynamics of traffic flow by embedding microscopic dynamics of automated vehicles into the mesoscopic traffic stream. This model could be used to describe Lagrangian traffic flow actuation.

Example 5.6.3 (mesoscopic traffic flow actuation using automated vehicles)**.**

$$q_t(t,x,v) + v\partial_x q(t,x,v) = -\partial_v \left(\left(\mathcal{H}[q(t), y(t), \dot{y}(t)](x,v) \right) q(t,x,v) \right) \tag{5.30}$$

$$q(0,x,v) = q_0(x,v), \quad (x,v) \in \mathbb{R} \times \mathbb{R}_{>0} \tag{5.31}$$

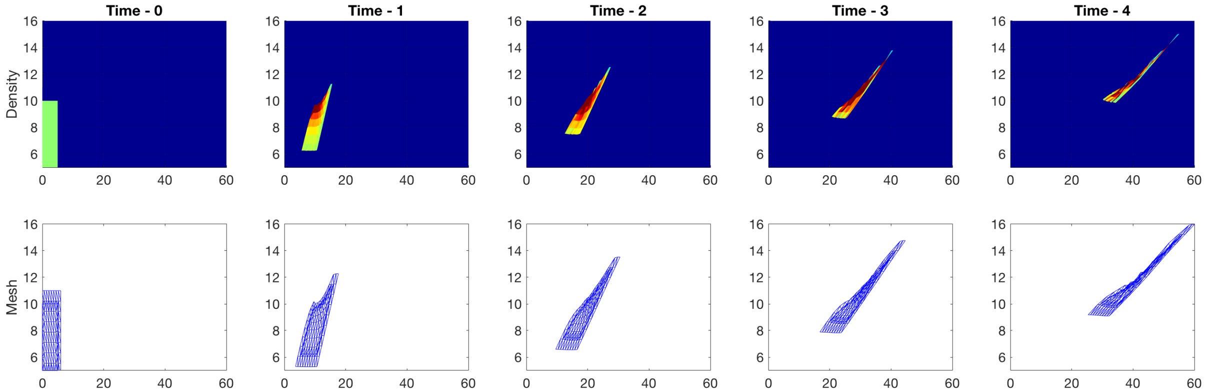


Figure 5.8: Mesh deformation and density concentration overtime. Numerical solution is shown in the top row, and underlying mesh is shown in the lower row.

with the nonlocal operators a defined in

$$\mathcal{H}[q,y,\dot{y}](x,v) = \alpha\mathcal{H}_1[q](x,v) + \beta\mathcal{H}_1^a[y](x,v) + \gamma\mathcal{H}_2^a[y,\dot{y}](x,v) \tag{5.32}$$

$$\mathcal{H}_1[q](x,v) = a - a\left(\frac{v}{v_0}\right)^{\delta} - a\int_{\mathbb{R}}\int_{\mathbb{R}_{>0}} h(x-\tilde{x})q(\tilde{x},\tilde{v})\left(\frac{s_0 + vT + \frac{v(v-\tilde{v})}{\sqrt{2ab}}}{\tilde{x}-x-l}\right)^2 \mathrm{d}\tilde{v}\,\mathrm{d}\tilde{x}. \tag{5.33}$$

$$\mathcal{H}_1^a[y](x,v) \coloneqq h(x-y)\left(V(y-x)-v\right), \tag{5.34}$$

$$\mathcal{H}_2^a[y,w](x,v) \coloneqq h(x-y)\cdot(w-v) \tag{5.35}$$

$$V(x) \coloneqq V_{\max}\frac{\tanh\left(\frac{x}{d_0}-2\right)+\tanh(2)}{1+\tanh(2)} \tag{5.36}$$

$$h(x) \coloneqq \begin{cases} \mathrm{e}^{-\frac{1}{\left(\frac{\varepsilon_0}{2}\right)^2-\left(-x-\frac{\varepsilon_0}{2}\right)^2}}, & \textit{if } -\varepsilon_0 < x < 0, \\ 0, & \textit{otherwise.} \end{cases} \tag{5.37}$$

coupled with

$$\ddot{y}(t) = \mathcal{H}[q(t),y(t),\dot{y}(t)](y(t),\dot{y}(t)) + u(t) \qquad t\in[0,T] \tag{5.38}$$

$$y(0) = y_0 \tag{5.39}$$

$$\dot{y}(0) = \dot{y}_0. \tag{5.40}$$

where v_o, δ, s_o, T, a, b, l, ε_o, α, β, γ, ε_0 *and* d_0 *are tuning parameters.*

A numerical solution of the system is shown in Figure 5.10. The top row of this figure shows the effect of three autonomous vehicles driving a slower speed diverging from the rest

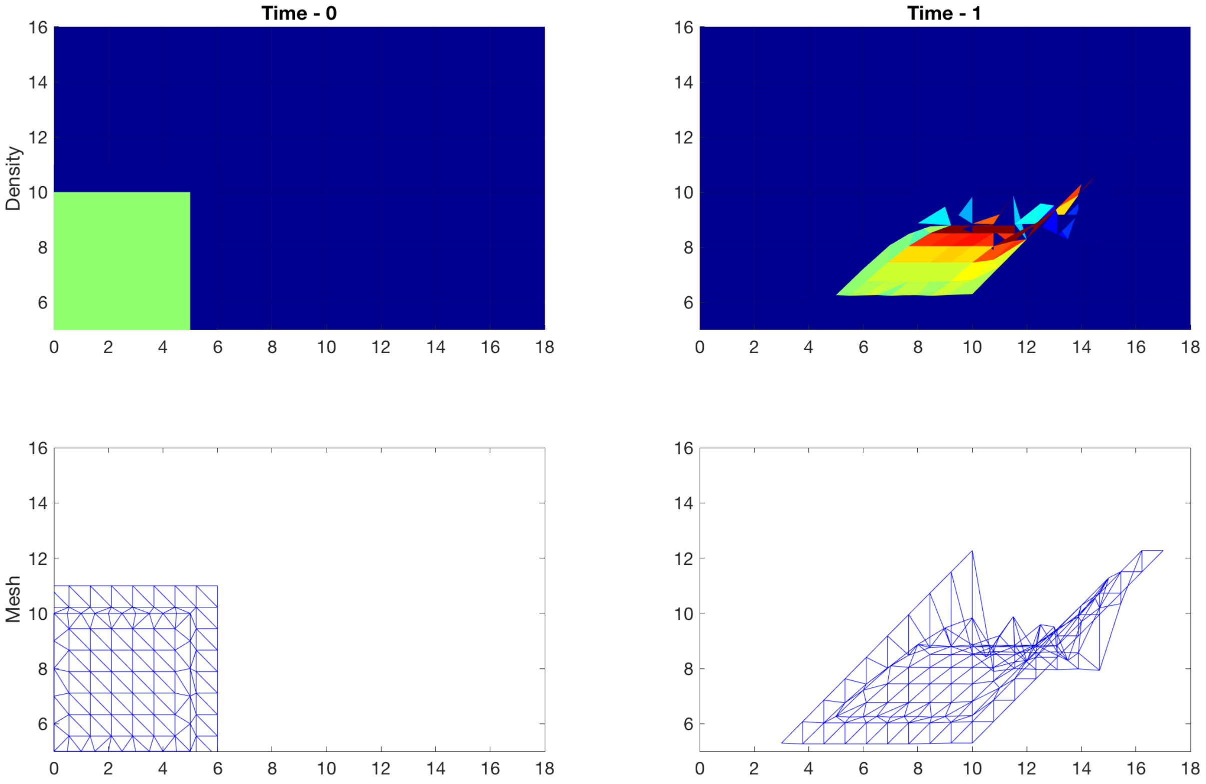


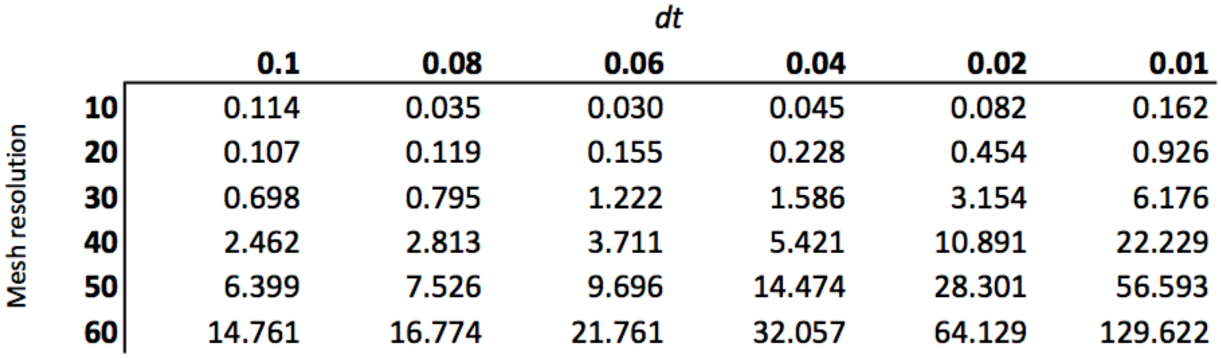
Figure 5.9: Triangle overlapping because of a too large time step δ_t. Numerical solution is shown in the top row, and underlying mesh is shown in the lower row.

of the stream. This could be compared to the standard traffic stream without actuation in the lower row of the same figure. The autonomous vehicles cause congestion behind them.

A replication of the numerical examples shown in Traffic flow model section is shown in Figure 5.11 and in Figure 5.12. Stronger effect for the autonomous vehicles can be observed in Figure 5.11 where the stream slows down while interacting with the autonomous vehicles. The same faster stream is slowing down in Figure 5.12 primarily due the slower traffic ahead of it. This makes it drive more consistently with the autonomous vehicles and hence autonomous vehicle effects are less significant. Again, solution robustness to mesh resolution and time step can be noted.

Computational time to solve this model is shown in Table 5.3. The system is solved for time horizon of one with an initial datum covering $[0, 20]^2$ domain similar to the setup presented in the earlier section. Computational complexity scales approximately linearly with number of simulated frames ($\frac{1}{dt}$) and quadratically in the number of triangles in the mesh as shown in Figure 5.13 and Figure 5.14. It can be observed from these tables and figures that the computational complexity of the ODE coupling is negligible compared to the computational complexity of the core PDE model as presented in the earlier section. Although not the main subject of this chapter, a detailed study of the computational complexity of the ODE coupling would require proper scaling of the ODE system as well, which was neglected here.

Table 5.3: Computational time in seconds to numerically solve the model in Example 5.6.3.



	dt					
Mesh resolution	**0.1**	**0.08**	**0.06**	**0.04**	**0.02**	**0.01**
10	0.114	0.035	0.030	0.045	0.082	0.162
20	0.107	0.119	0.155	0.228	0.454	0.926
30	0.698	0.795	1.222	1.586	3.154	6.176
40	2.462	2.813	3.711	5.421	10.891	22.229
50	6.399	7.526	9.696	14.474	28.301	56.593
60	14.761	16.774	21.761	32.057	64.129	129.622

Nondissipative property

The solver presented in this chapter is nondissipative; it can numerical solve for models with discontinuities accurately. This maintains structure and validity of the solution for longer time horizons. To illustrate this property, this section compares the approach in the chapter to the classical solver presented in [102]. In Figure 5.15, an initial datum with rectangular support is used to solve the model in Example 5.6.3. The top row shows numerical solution generated using our approach at several time steps, while the lower row shows a numerical solution using the classical solver at the same time steps. The initial datum used for the classical solver is moved by a small margin from the boundary of the domain because of ghost cells.

The solver in this chapter inherently tracks discontinuities accurately compared to classical viscosity approximation methods, which on the other hand inherently smooth solutions over time. Classical solver smoothing effects could be reduced by tuning solver parameters, which is a challenging exercise, yet the effect is expected to persist. This is because the CFL condition must hold and requiring a significantly smaller time steps, and in the case of this specific example, the CFL must be picked exactly equal to one for the discontinuities to move with the proper velocity, which is challenging since the velocity field is space-velocity dependent (as per $\mathcal{H}$).

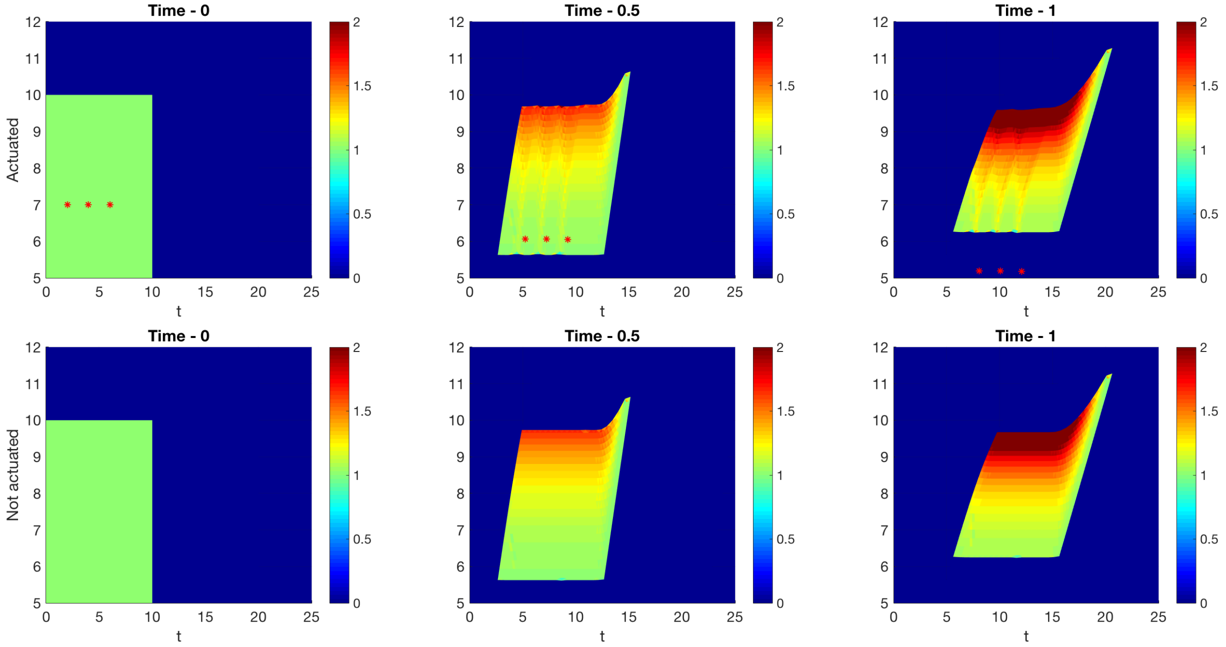


Figure 5.10: Numerical solution to actuated traffic model presented in Example 5.6.3 and non-actuated traffic model presented in Example 5.6.2. Red asterisks represent actuated vehicles.

5.7 Conclusion

This chapter presented an L^1 conserving numerical solver for nonlocal conservation laws. The solver supports tracking of discontinuities and is robust to discretization resolution. The chapter presented the theoretical basis for the solver along with a vectorized representation of the core equations for implementation, and were used to develop the solver algorithm. Numerical traffic flow modeling and Lagrangian traffic actuation examples were presented.

The solver's robustness to discretization resolution while maintaining properties of the solution makes it suitable for compute extensive methods such as numerical analysis and optimal control. Moreover, optimal control using adjoint equation method could further benefit from the re-utilization of mesh evolution forwards in time to improve accuracy and scalability while solving backwards in time. The solver can support parallesim—by design—to solve system models such as extensions to traffic models from one link to models of several road links.

The solver introduces four types of error; triangulation of initial datum, approximation of the nonlocal term, triangle geometry approximation over time, and approximate mesh restorations. Some error introduced by the solver could be remedied by introducing more complexity or prior analysis of model equations. For instance, the nonlocal convolution could be computed explicitly, which would reduce the error introduced to the solution, and would

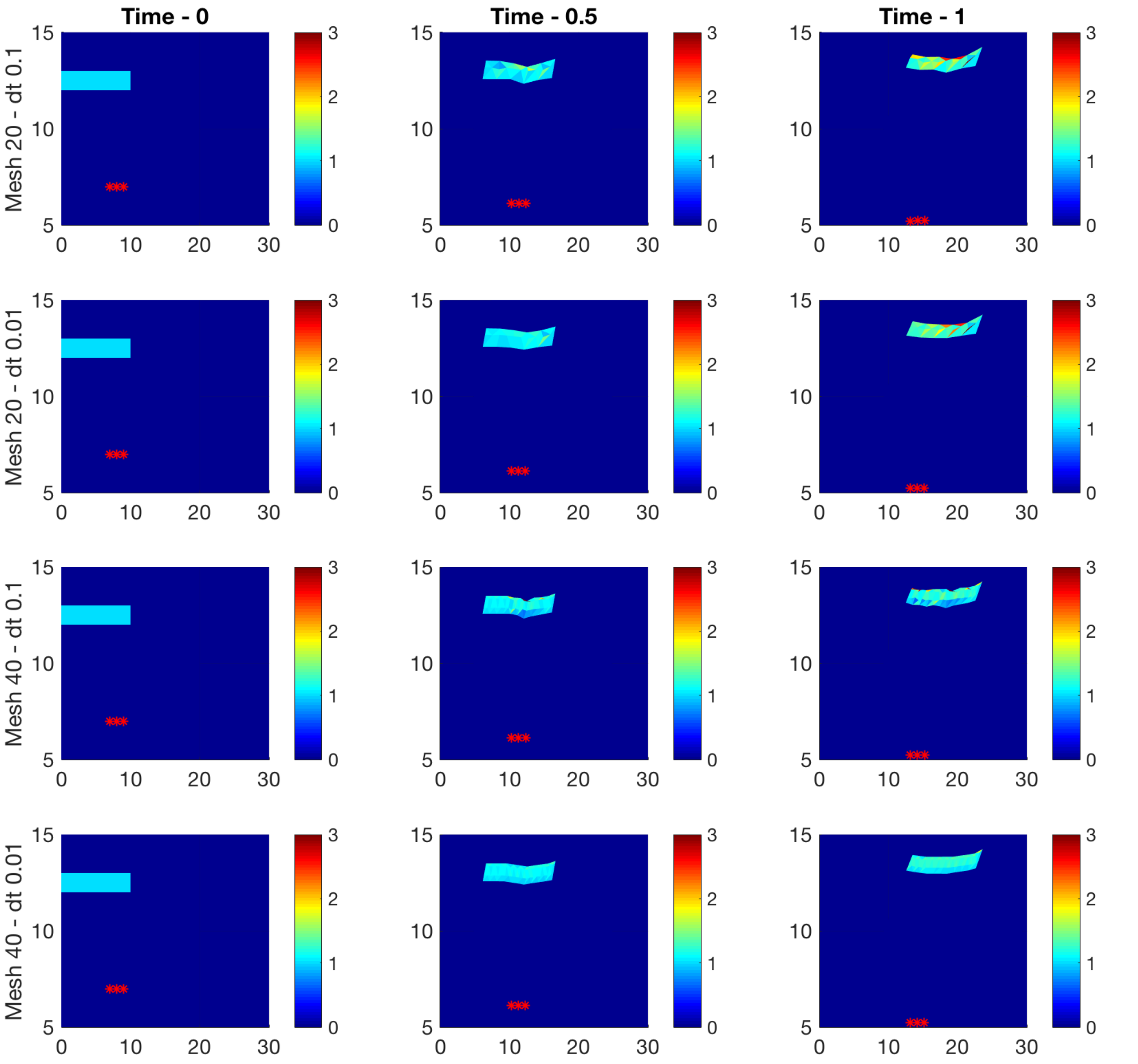


Figure 5.11: Numerical solution to model presented in Example 5.6.2 with a rectangular initial datum and three autonomous vehicles embedded into the stream. Red asterisks represent actuated vehicles.

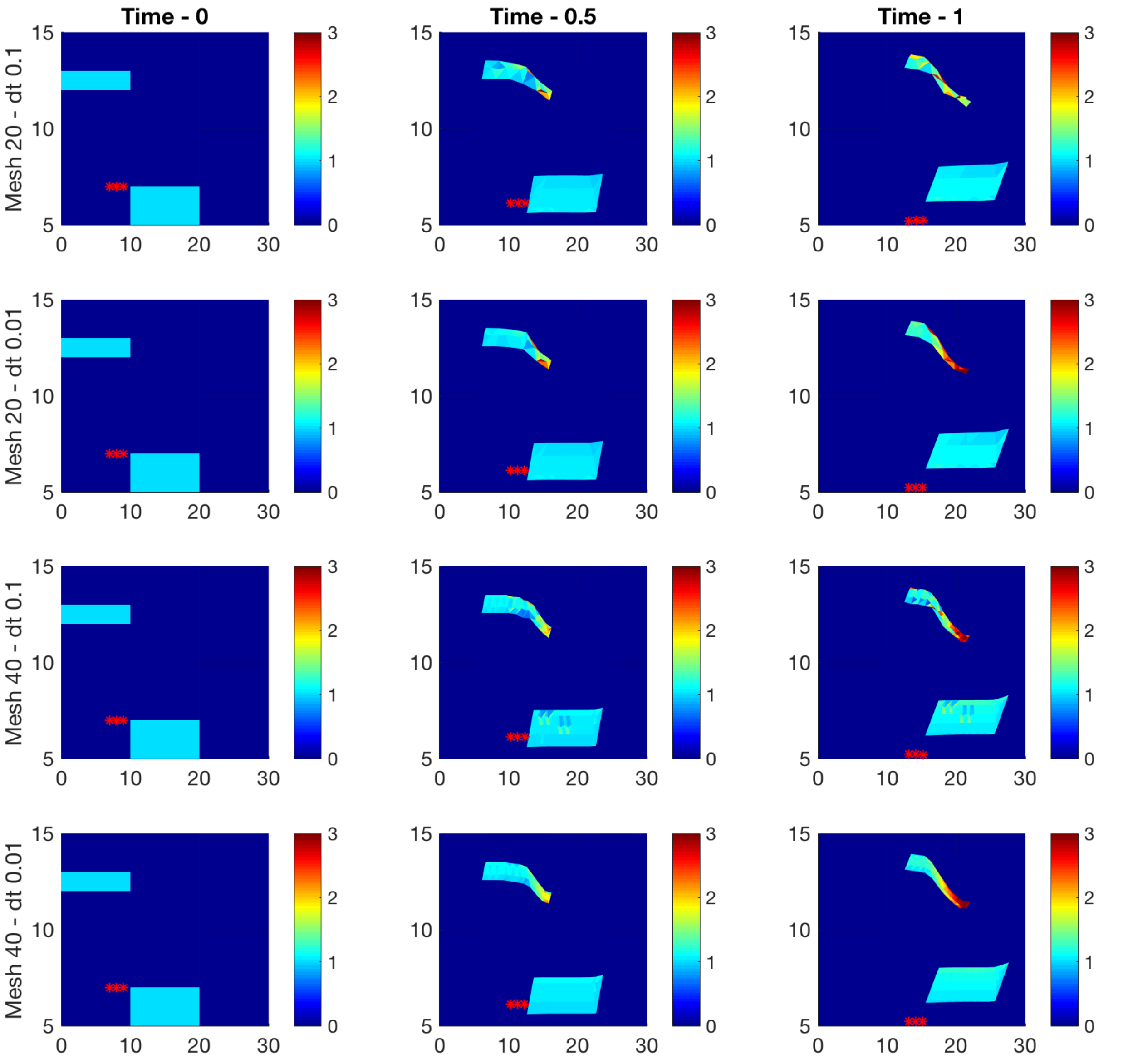


Figure 5.12: Numerical solution to model presented in Example 5.6.3 with two interacting rectangular regions in the initial datum and three autonomous vehicles embedded into the stream. Red asterisks represent actuated vehicles.

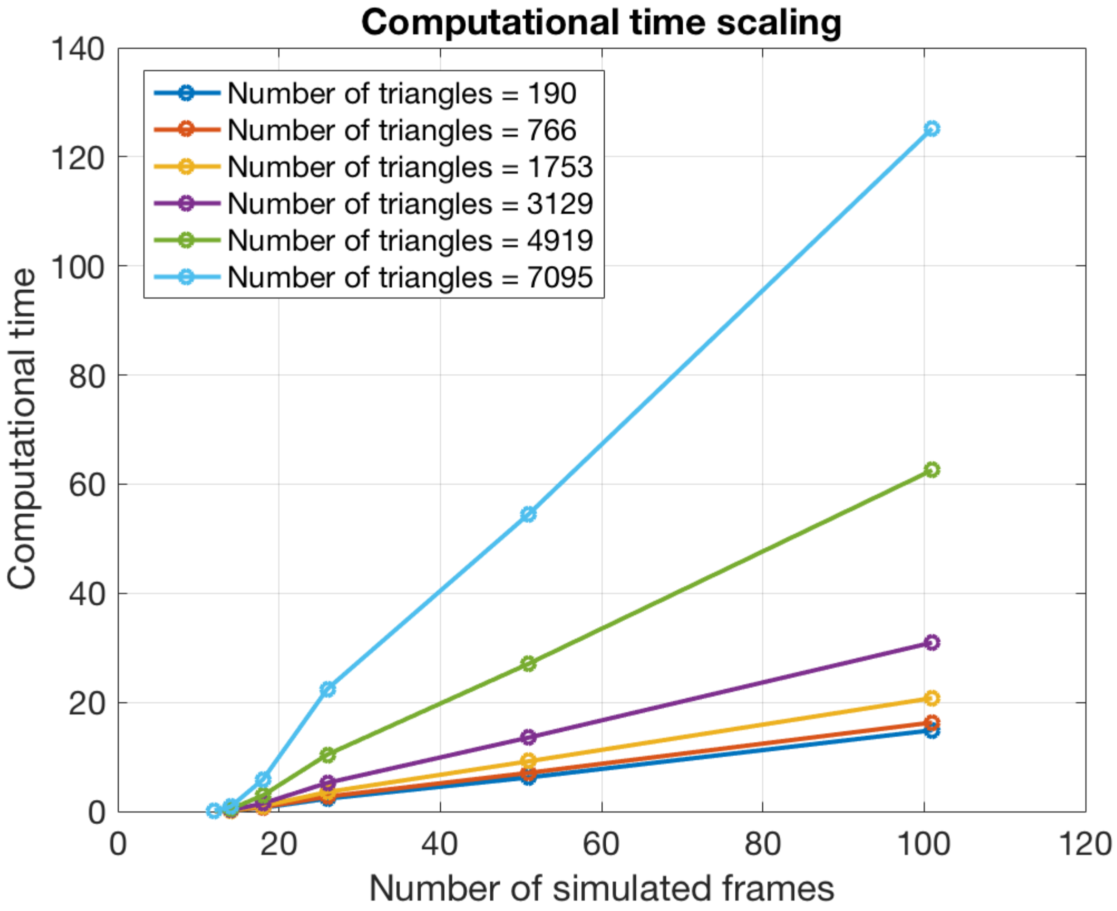


Figure 5.13: Computational time scaling as a function of the number simulated time frames for several mesh resolutions.

further robustify the solver to triangulation resolution.

The numerical solution approach presented here depends on an evolving in time mesh. The mesh deforms and could degenerate over time potentially degrading the accuracy of the solution. We introduced a simple algorithm that could be used to restore the solution. More detailed algorithms could be implemented to ensure higher accuracy tracking of mesh quality and mesh restoration [129]–[133].

The solver could be further improved by extensions to support for source terms and boundary value problems, the implementation of higher order numerical integrators, and a study of parallelization. Careful convergence analysis is still necessary to support the validity of the solver; this would require a study of how the solution changes as $\Delta \to 0$ and $\delta_t \to 0$ and at which precision it would converge to the real solution. By construction, and based on the supporting theory, we speculate that with the solver presented in this chapter has a

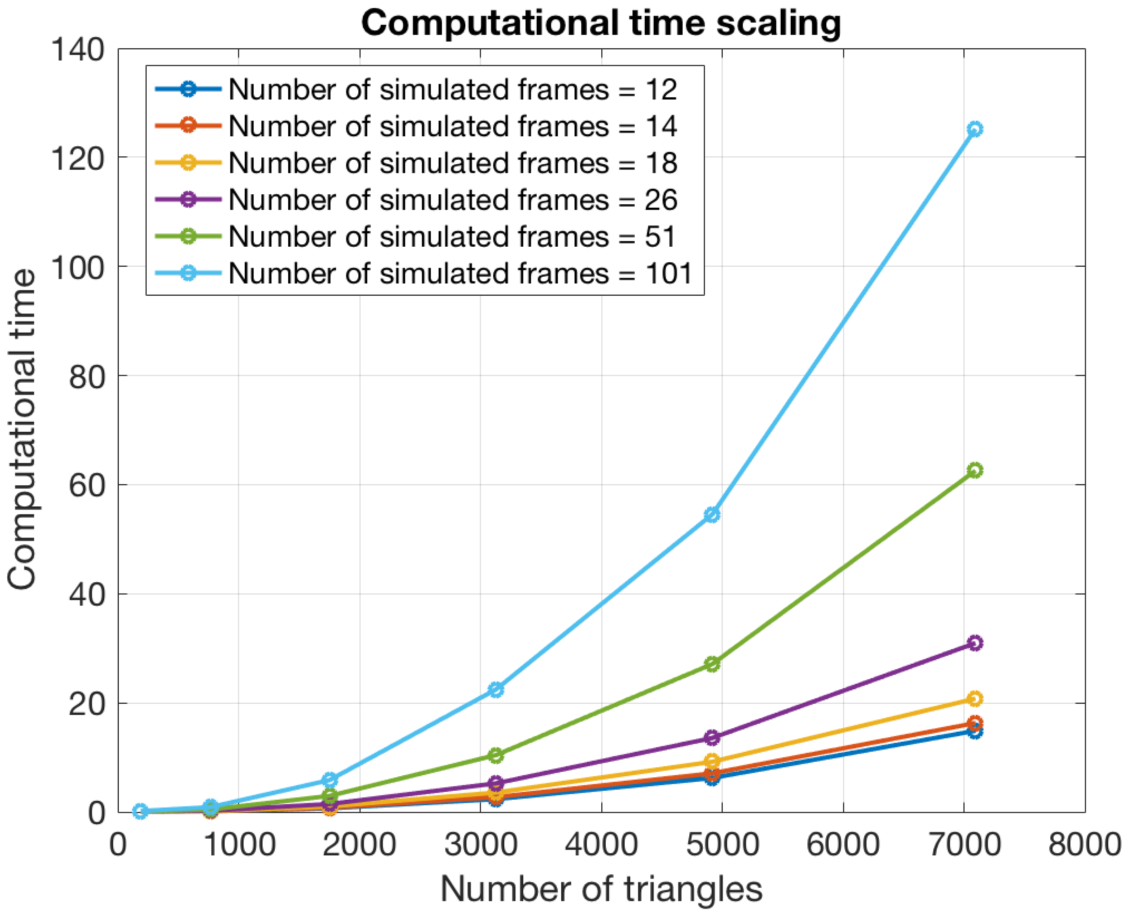


Figure 5.14: Computational time scaling as a function of mesh resolution for several time step settings.

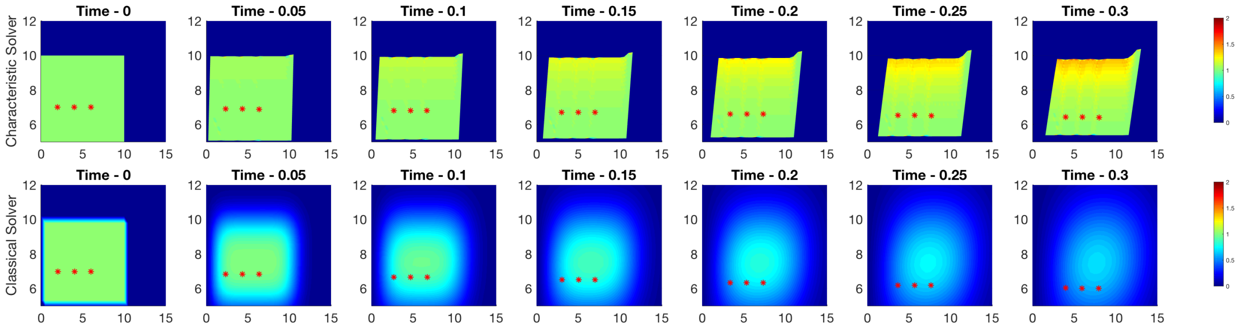


Figure 5.15: Numerical solution to actuated traffic model presented in Example 5.6.3 using the solver presented in this chapter and the solver presented in [102]. Red asterisks represent actuated vehicles.

linear convergence in δ_t and mesh resolution.

Chapter 6

Conclusion and future steps

6.1 Introduction

Transportation system consumes about a third of all the energy consumed in the world, and a third of which is consumed by trucks [2], [11]. Truck automation could potentially lead to fuel consumption reduction [17] and flow capacity improvement [18]. Long haul heavy duty trucks spend significant amount of time driving in freeways, a relatively simple driving environment compared to urban driving environments. This makes autonomous long haul trucks amongst the most likely vehicles to become the first to mature for full deployment. However, heavy duty trucks are highly complex compared to other ground vehicles and require highly specialized engineering tools to study and automate [16], [33]–[37]. Black-box computational tools help abstract the complexity and have improved accuracy in many modern applications including vehicle dynamics and control [9], [10]. For heavy duty trucks, this thesis developed and demonstrated a process for microscopic modeling and control using deep learning and deep reinforcement learning. To incorporate background vehicular traffic on the road for truck automation, the thesis considered multi-scale mean-field models. A structure preserving numerical solver was developed to allow for the study of these models and for the development of optimal control schemes.

6.2 Discussion and open problems

Heavy duty trucks are complex vehicles that are designed and built for specific mission requirements. Any of these trucks could be equipped from a wide selection of vehicle components with a significantly wide spectrum of operating dynamics and performances. Driving a heavy duty truck is an equally complex task. Expert human drivers must be well educated and trained about the specific truck they are about to drive and operate.

A human driver must optimize in real-time for factors such as truck dynamics and driving performance; road, truck, and payload safety; truck operation economics; truck driving law constraints; mission constraints; in addition to background traffic on the road. While doing so, they must continuously monitor and make complex decisions. For instance, to decelerate, they must decide if they should apply any brake, and if so, they must decide between several brake system options that might be available to them, each with a different performance and economics characteristics. They must take into account the surrounding traffic situation and road topography to make time optimal decisions for brake application timing and duration.

Automation of these tasks historically required advanced expertise and advanced theoretical and applied tools. Automation of longitudinal driving require an expert level understanding of the specific maneuver (such speed regulation or leader following) of interest. Maneuver specifications inform a coupled modeling, control design, and validation processes. Mathematical models are custom built to take into account the significant dynamics of the truck important to achieve the maneuver, the theoretical control framework requirements and limitations, and practical aspects such as available information, details, and accessibility to specific truck components.

Common truck mechanical configurations and commonly utilized components are modeled for high fidelity simulation frameworks such as TruckSIM. These frameworks require expert level knowledge of vehicle mechanics and their mechanical configurations to operate. Modeling of a specific truck of interest into these frameworks often require more than calibration of model components. Extensions and custom built models may be required to produce sufficiently accurate models. An example of these often required extensions include specialized types of retarders and custom built service brakes.

Deep and deep reinforcement learning approach. Chapter 3, Microscopic multi-vehicle modeling and coordination using deep learning and deep reinforcement learning, and Chapter 4, Vehicle actuation using deep reinforcement learning, introduced a process for microscopic modeling and control of heavy duty trucks using deep learning and deep reinforcement learning. Deep learning and deep reinforcement learning are frameworks that have proved effectiveness to abstract complex dynamical details. These two chapter develop a process to abstract required expertise in trucks and dynamical systems to streamline the development process of truck models and controllers.

With this process as an alternative approach, truck models and controller could be developed with significantly higher efficiency; (1) drive the truck to collect information about its dynamics in form of data, (2) pass the data to a black-box deep learning system to produce a fitted simulation model of the truck, (3) pass the simulation model to a black-box deep reinforcement learning system to produce simulation based optimal controllers. For the three trucks prominently used for validation in this thesis, and the examples of cruise controllers and cooperative adaptive cruise controllers, this process produced models and controllers that are ready to ship within hours from start to finish; including time for data collection, training, and packaging. This could be compared to classical approaches, which are often iterative, and would require days or weeks to calibrate and tune.

The work presented in these two chapters is however just the first step to simplify the process of truck automation system development. The method as described in this manuscript produce models that are over-fit to the specific truck configuration for which that data was collected, and produce controllers that are over-fit to the specific truck models and for the specific control task used for training. This issue appears for any single truck and between multiple trucks.

For a single truck, configuration was assumed static. However, a truck could change configuration as it operates; this could happen for instance when a trailer is attached or removed. And between trucks and control tasks, no transfer learning was considered. This is an inefficient utilization of data, compute power for training, and could eventually lead into a difficult to manage collections of data, models, environments, etc.

Moreover, the process considered here does not utilize existing literature on physics based vehicle modeling and control. For instance, gravity models are well understood by the classical theory of vehicle dynamics. Incorporating gravity models into the deep modeling and control process could significantly reduce the amount of required data for modeling and the required scenarios for control learning.

The process is also challenged by many practical details, some of which still require expert level knowledge to address. For instance, asymmetric data access and control access is a common problem for custom built vehicle automation platforms. The service brake of the Volvo truck presented in this manuscript accepts deceleration in m/s^2 as a control command, which is then converted internally (using propriety controllers) to raw brake cylinder actuation signals. Only brake pedal deflection is accessible for data collection from open loop human driving sessions. This required either data collection from a scheduled automated sessions, which can be challenging, or by manipulating the data to approximate expected signals by the truck.

In this research, we used heuristics to specify required amount of data and scenarios to consider for data collection and for simulation based control design. A more formal investigation of necessary and sufficient data and scenarios is still required. Moreover, these chapters conducted only a brief comparative analysis to classical control approaches. A more extensive comparative analysis is still necessary, along with more extensive performance and robustness analysis. Moreover, black-box models such as deep learning and deep reinforcement learning controllers offer abstraction of details and performance improvement at the expense of guarantees that come with classical methods. While not the subject of this thesis, this challenge is also still open for further investigation.

Numerical solver for multi-scale mean-field models. Chapter 5, Multi-scale vehicle and traffic flow modeling and actuation using mean-field models, introduced multi-scale mean-field modeling of microscopic automated vehicles with mesoscopic background traffic. Mean-field theory provides a formal framework for theoretical study of mesoscopic models given microscopic descriptions of the constituent agents. The literature on numerical solvers for these models is still sparse, which is otherwise necessary to incorporate these models into computational frameworks.

The numerical solver developed maintains structure and preserves discontinuities allowing for higher accuracy of numerical solutions for longer time horizons. This could be compared to classical approaches for numerical solvers of PDEs, which use viscosity approximation and smooth solutions. Smoothing could be reduced by optimizing solver parameter. However, this is not always easily achievable, such as for the case of the examples presented in this manuscript, for which the velocity of the discontinuity is space-velocity dependent.

With the approach of this solver, mesh quality is tracked as a CFL type condition, which is otherwise difficult to define and tune. Euler time discretization could cause triangles to overlap, and model dynamics could cause individual triangles to degenerate. These conditions could be easily defined, implemented, verified, and corrected for.

With this approach, only the non-zero regions of the initial datum must be meshed and solver for, for the remainder of the time horizon. This reduces the computational complexity compared to solvers that require meshing and solving for all parts of the domain where mass could evolve into.

Moreover, numerical approximations generated using this numerical solver are robust to discretization resolution (both time and mesh). This makes it suitable for compute intensive applications such as optimal control strategies using adjoint equations. Computational efficiency of the optimizer could be managed by employing adaptive resolution strategies.

This manuscript validated this numerical solver using a model for which an explicit solution exists. The model used is, however, primitive. More complex models with explicit solutions must be used for more rigorous validation. Moreover, robustness to discretization resolution is only shown with brief numerical examples. A more formal theoretical convergence analysis is still required.

We utilized several simplifying assumptions to develop the numerical solver presented here. For instance, the initial datum was assumed triangle-piece-wise constant, we assumed triangles maintain geometry, and we employed simple first order Euler time discretization. While the overall numerical approximations remain valid, accuracy could be improved by employing higher order methods.

The approach of this numerical solver utilizes a moving in time mesh that can degenerate. More investigation is still required to refine the algorithm presented here to correct for this degeneration. The algorithm does not clearly answer questions (or make simplifying assumptions) such as the choice of a proper interpolation function, the approach for adaptive remeshing (denser areas maintain denser triangulation), accuracy loss to these corrective steps, computational complexity to conduct these steps, etc.

6.3 Future steps

Automated vehicles operate within a larger complex context of the transportation systems while serving mobility missions. Development of fully autonomous vehicles of the future would require a comprehensive investigation and consideration of at least the vehicle, the system where it operates, the mission it serves, along with the human beneficiary. Aspects of fully autonomous vehicular systems have been considered for ground transportation and mobility systems [134]–[147], air traffic systems [148]–[150], and warehouses [151], [152].

The literature on fully autonomous trucking systems is still sparse. Enriching this literature would require the development of fully autonomous truck system modeling framework. Such framework would detail trucks as primary vehicles, the transportation system as the context of operation, the logistic system for mission specification. This modeling framework could then be extended to develop an automation framework that brings together the wealth of literature on strategies to control and optimize the performance of the system. Such modeling and automation frameworks could help study in detail the opportunities and the potential for future fully autonomous ground freight.

Bibliography

[1] *How much time do americans spend behind the wheel? volpe national transportation systems center*, 2019. [Online]. Available: https://www.volpe.dot.gov/news/how-much-time-do-americans-spend-behind-wheel.

[2] *Energy use for transportation. u.s. energy information administration (eia)*, 2019. [Online]. Available: https://www.eia.gov/energyexplained/use-of-energy/transportation.php.

[3] *Highway and road expenditures. urban institute*, 2019. [Online]. Available: https://www.urban.org/policy-centers/cross-center-initiatives/state-and-local-finance-initiative/state-and-local-backgrounders/highway-and-road-expenditures.

[4] K. Boriboonsomsin, M. J. Barth, W. Zhu, and A. Vu, "Eco-routing navigation system based on multisource historical and real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1694–1704, 2012.

[5] D. J. Fagnant and K. M. Kockelman, "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 1–13, 2014.

[6] S. Le Vine, O. Adamou, and J. Polak, "Predicting new forms of activity/mobility patterns enabled by shared-mobility services through a needs-based stated-response method: Case study of grocery shopping," *Transport Policy*, vol. 32, pp. 60–68, 2014.

[7] A. M. Robitaille, J. Methipara, and L. Zhang, "Effectiveness and equity of vehicle mileage fee at federal and state levels," *Transportation research record*, vol. 2221, no. 1, pp. 27–38, 2011.

[8] J. Firnkorn and M. Müller, "Selling mobility instead of cars: New business strategies of automakers and the impact on private vehicle holding," *Business Strategy and the environment*, vol. 21, no. 4, pp. 264–280, 2012.

[9] M. Da Lio, D. Bortoluzzi, and G. P. Rosati Papini, "Modelling longitudinal vehicle dynamics with neural networks," *Vehicle System Dynamics*, pp. 1–19, 2019.

[10] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, eaaw1975, 2019.

[11] *Transportation sector energy consumption. bureau of transportation statistics*, 2019. [Online]. Available: `https://www.eia.gov/outlooks/ieo/pdf/transportation.pdf`.

[12] *Freight shipments by mode. bureau of transportation statistics*, 2019. [Online]. Available: `https://bts.gov/topics/freight-transportation/freight-shipments-mode`.

[13] *2017 north american freight numbers. bureau of transportation statistics*, 2019. [Online]. Available: `https://www.bts.gov/newsroom/2017-north-american-freight-numbers`.

[14] X.-Y. Lu and J. K. Hedrick, "Heavy-duty vehicle modelling and longitudinal control," *Vehicle System Dynamics*, vol. 43, no. 9, pp. 653–669, 2005.

[15] X.-Y. Lu and S. Shladover, "Integrated acc and cacc development for heavy-duty truck partial automation," in *2017 American Control Conference (ACC)*, IEEE, 2017, pp. 4938–4945.

[16] X.-Y. Lu and J. K. Hedrick, "Heavy-duty vehicle modelling and longitudinal control," *Vehicle System Dynamics*, vol. 43, no. 9, pp. 653–669, 2005.

[17] B. McAuliffe, M. Lammert, X.-Y. Lu, S. Shladover, M.-D. Surcel, and A. Kailas, "Influences on energy savings of heavy trucks using cooperative adaptive cruise control," SAE Technical Paper, Tech. Rep., 2018.

[18] S. E. Shladover, D. Su, and X.-Y. Lu, "Impacts of cooperative adaptive cruise control on freeway traffic flow," *Transportation Research Record*, vol. 2324, no. 1, pp. 63–70, 2012.

[19] M. Schäfer, *Computational engineering: introduction to numerical methods.* Springer, 2006.

[20] B. Açıkmeşe, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[21] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[22] P. Kersaudy, B. Sudret, N. Varsier, O. Picon, and J. Wiart, "A new surrogate modeling technique combining kriging and polynomial chaos expansions–application to uncertainty analysis in computational dosimetry," *Journal of Computational Physics*, vol. 286, pp. 103–117, 2015.

[23] R. Tripathy and I. Bilionis, "Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification," *arXiv preprint arXiv:1802.00850*, 2018.

[24] W. A. Pruett and R. L. Hester, “The creation of surrogate models for fast estimation of complex model outcomes,” *PloS one*, vol. 11, no. 6, e0156574, 2016.

[25] F. Lamperti, A. Roventini, and A. Sani, “Agent-based model calibration using machine learning surrogates,” *Journal of Economic Dynamics and Control*, vol. 90, pp. 366–389, 2018.

[26] B. A. Furtado, “Machine learning simulates agent-based model,” *arXiv preprint arXiv:1712.04429*, 2017.

[27] A. Manik, K. Gopalakrishnan, A. Singh, and S. Yan, “Neural networks surrogate models for simulating payment risk in pavement construction,” *Journal of Civil Engineering and Management*, vol. 14, no. 4, pp. 235–240, 2008.

[28] N. Fouladinejad, N. Fouladinejad, M. K. Abdul Jalil, and J. Mohd Taib, “Development of a surrogate-based vehicle dynamic model to reduce computational delays in a driving simulator,” *Simulation*, vol. 92, no. 12, pp. 1087–1102, 2016.

[29] I. Wallach, M. Dzamba, and A. Heifets, “Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery,” *arXiv preprint arXiv:1510.02855*, 2015.

[30] J. Wang, H. Cao, J. Z. Zhang, and Y. Qi, “Computational protein design with deep learning neural networks,” *Scientific reports*, vol. 8, no. 1, p. 6349, 2018.

[31] S. van der Hoog, “Deep learning in (and of) agent-based models: A prospectus,” *arXiv preprint arXiv:1706.06302*, 2017.

[32] R. B. Parra-Galaviz, M. Castanón–Puga, C. Gaxiola-Pacheco, and D. Suarez, “Towards decision making systems from data ethnography for agent-based simulation using computational intelligence,”

[33] F. Lattemann, K. Neiss, S. Terwen, and T. Connolly, “The predictive cruise control–a system to reduce fuel consumption of heavy duty trucks,” *SAE transactions*, pp. 139–146, 2004.

[34] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager, “Mixed-integer nmpc for predictive cruise control of heavy-duty trucks,” in *2013 European Control Conference (ECC)*, IEEE, 2013, pp. 4118–4123.

[35] H. S. Bae and J. C. Gerdes, “Parameter estimation and command modification for longitudinal control of heavy vehicles,” 2003.

[36] A. Vahidi, M. Druzhinina, A. Stefanopoulou, and H. Peng, “Simultaneous mass and time-varying grade estimation for heavy-duty vehicles,” in *Proceedings of the 2003 American Control Conference, 2003.*, IEEE, vol. 6, 2003, pp. 4951–4956.

[37] M. Druzhinina and A. Stefanopoulou, “Speed control experiments for commercial heavy vehicles with coordinated friction and engine compression brakes,” in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, IEEE, vol. 3, 2002, pp. 2546–2551.

[38] R. Rajamani, *Vehicle dynamics and control.* Springer Science & Business Media, 2011.

[39] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, 2014.

[40] V. Zhang, S. M. Thornton, and J. C. Gerdes, "Tire modeling to enable model predictive control of automated vehicles from standstill to the limits of handling," in *14th International Symposium on Advanced Vehicle Control (AVEC 2018)*, 2018.

[41] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "Mpc-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2, pp. 265–291, 2005.

[42] L. Ng, "Reinforcement learning of dynamic collaborative driving," 2008.

[43] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*, Springer, 2006, pp. 363–372.

[44] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015.

[45] N. Mohajerin, M. Mozifian, and S. Waslander, "Deep learning a quadrotor dynamic model for multi-step prediction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[46] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," *arXiv preprint arXiv:1610.05863*, 2016.

[47] S. Albeaik, F.-C. Chou, X.-Y. Lu, and A. M. Bayen, "Deep truck: A deep neural network model for longitudinal dynamics of heavy duty trucks," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 4158–4163.

[48] S. Albeaik, T. Wu, G. Vurimi, X.-Y. Lu, and A. M. Bayen, "Longitudinal deep truck," in-review 2020.

[49] S. Albeaik, T. Wu, G. Vurimi, F.-C. Chou, X.-Y. Lu, and A. M. Bayen, "Deep cruise control," in-review 2020.

[50] T. Sandberg, *Heavy truck modeling for fuel consumption simulations and measurements.* Univ., 2001.

[51] A. Al Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *13th International IEEE Conference on Intelligent Transportation Systems*, IEEE, 2010, pp. 306–311.

[52] M. P. Lammert, A. Duran, J. Diez, K. Burton, and A. Nicholson, "Effect of platooning on fuel consumption of class 8 vehicles over a range of speeds, following distances, and mass," *SAE International Journal of Commercial Vehicles*, vol. 7, no. 2014-01-2438, pp. 626–639, 2014.

[53] J. Smith, R. Mihelic, B. Gifford, and M. Ellis, "Aerodynamic impact of tractor-trailer in drafting configuration," *SAE International Journal of Commercial Vehicles*, vol. 7, no. 2014-01-2436, pp. 619–625, 2014.

[54] D. Norrby, "A cfd study of the aerodynamic effects of platooning trucks," PhD thesis, Master's thesis, KTH Royal Institute of Technology, Stockholm, 2014.

[55] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, "Cooperative adaptive cruise control: Definitions and operating concepts," *Transportation Research Record*, vol. 2489, no. 1, pp. 145–152, 2015.

[56] A. Alam, B. Besselink, V. Turri, J. Mårtensson, and K. H. Johansson, "Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency," *IEEE Control Systems Magazine*, vol. 35, no. 6, pp. 34–56, 2015.

[57] D. Swaroop, J. K. Hedrick, C. Chien, and P. Ioannou, "A comparision of spacing and headway control laws for automatically controlled vehicles1," *Vehicle system dynamics*, vol. 23, no. 1, pp. 597–625, 1994.

[58] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 4, pp. 695–708, 2000.

[59] R. Rajamani and S. E. Shladover, "An experimental comparative study of autonomous and co-operative vehicle-follower control systems," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 15–31, 2001.

[60] T. Robinson, E. Chan, and E. Coelingh, "Operating platoons on public motorways: An introduction to the sartre platooning programme," in *17th world congress on intelligent transport systems*, vol. 1, 2010, p. 12.

[61] H. Fritz, A. Gern, H. Schiemenz, and C. Bonnet, "Chauffeur assistant: A driver assistance system for commercial vehicles based on fusion of advanced acc and lane keeping," in *IEEE Intelligent Vehicles Symposium, 2004*, IEEE, 2004, pp. 495–500.

[62] A. Alam, B. Besselink, V. Turri, J. Martensson, and K. H. Johansson, "Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency," *IEEE Control Systems*, vol. 35, no. 6, pp. 34–56, 2015.

[63] L. Jin, M. Čičić, S. Amin, and K. H. Johansson, "Modeling the impact of vehicle platooning on highway congestion: A fluid queuing approach," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, 2018, pp. 237–246.

[64] H. Liu, X. Kan, S. E. Shladover, X.-Y. Lu, and R. E. Ferlis, "Impact of cooperative adaptive cruise control on multilane freeway merge capacity," *Journal of Intelligent Transportation Systems*, vol. 22, no. 3, pp. 263–275, 2018.

[65] V. Milanés and S. E. Shladover, “Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data,” *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 285–300, 2014.

[66] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, “Cooperative adaptive cruise control in real traffic situations,” *IEEE Transactions on intelligent transportation systems*, vol. 15, no. 1, pp. 296–305, 2013.

[67] D. Swaroop and J. K. Hedrick, “String stability of interconnected systems,” *IEEE transactions on automatic control*, vol. 41, no. 3, pp. 349–357, 1996.

[68] X.-Y. Lu and J. K. Hedrick, “Practical string stability for longitudinal control of automated vehicles,” *Vehicle System Dynamics*, vol. 41, pp. 577–586, 2004.

[69] V. Turri, B. Besselink, and K. H. Johansson, “Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 12–28, 2016.

[70] S. Van De Hoef, K. H. Johansson, and D. V. Dimarogonas, “Fuel-optimal centralized coordination of truck platooning based on shortest paths,” in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 3740–3745.

[71] S. Van De Hoef, K. H. Johansson, and D. V. Dimarogonas, “Coordinating truck platooning by clustering pairwise fuel-optimal plans,” in *2015 IEEE 18th international conference on intelligent transportation systems*, IEEE, 2015, pp. 408–415.

[72] S. van de Hoef, K. H. Johansson, and D. V. Dimarogonas, “Computing feasible vehicle platooning opportunities for transport assignments,” *IFAC-papersonline*, vol. 49, no. 3, pp. 43–48, 2016.

[73] S. E. Shladover, X.-Y. Lu, B. Song, S. Dickey, C. Nowakowski, A. Howell, F. Bu, D. Marco, H.-S. Tan, and D. Nelson, “Demonstration of automated heavy-duty vehicles,” 2006.

[74] M. W. Sayers and S. M. Riley, “Modeling assumptions for realistic multibody simulations of the yaw and roll behavior of heavy trucks,” SAE Technical Paper, Tech. Rep., 1996.

[75] *TruckSim Quick Start Guide*. 2016.

[76] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[77] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[78] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Foundations and trends in signal processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[79] S. Haykin, “A comprehensive foundation,”

[80] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[81] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.."

[82] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[83] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[84] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[85] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen, "Emergent behaviors in mixed-autonomy traffic," in *Conference on Robot Learning*, 2017, pp. 398–407.

[86] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[87] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[88] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," *arXiv preprint arXiv:1002.4908*, 2010.

[89] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa, "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam," *New journal of physics*, vol. 10, no. 3, p. 033 001, 2008.

[90] S.-i. Tadaki, M. Kikuchi, M. Fukui, A. Nakayama, K. Nishinari, A. Shibata, Y. Sugiyama, T. Yosida, and S. Yukawa, "Phase transition in traffic jam experiment on a circuit," *New Journal of Physics*, vol. 15, no. 10, p. 103 034, 2013.

[91] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

[92] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *arXiv preprint arXiv:1710.05465*, p. 10, 2017.

[93] A. R. Kreidieh, C. Wu, and A. M. Bayen, “Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 1475–1480.

[94] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen, “Simulation to scaled city: Zero-shot policy transfer for traffic control via autonomous vehicles,” in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 291–300.

[95] H. J. Payne, “Model of freeway traffic and control,” *Mathematical Model of Public System*, pp. 51–61, 1971.

[96] A. Aw and M. Rascle, “Resurrection of” second order” models of traffic flow,” *SIAM journal on applied mathematics*, vol. 60, no. 3, pp. 916–938, 2000.

[97] I. Yperman, S. Logghe, and B. Immers, “The link transmission model: An efficient implementation of the kinematic wave theory in traffic networks,” in *Proceedings of the 10th EWGT Meeting*, Poznan Poland, 2005, pp. 122–127.

[98] M. Garavello and B. Piccoli, *Traffic flow on networks.* American institute of mathematical sciences Springfield, 2006, vol. 1.

[99] X. Gong, B. Piccoli, and G. Visconti, “Mean-field limit of a hybrid system for multilane multi-class traffic,” *(Submitted)*, 2020.

[100] E. Cristiani, B. Piccoli, and A. Tosin, *Multiscale modeling of pedestrian dynamics.* Springer, 2014, vol. 12.

[101] ——, “Multiscale modeling of granular flows with application to crowd dynamics,” *Multiscale Modeling & Simulation*, vol. 9, no. 1, pp. 155–182, 2011.

[102] A. Aggarwal, R. M. Colombo, and P. Goatin, “Nonlocal systems of conservation laws in several space dimensions,” *SIAM Journal on Numerical Analysis*, vol. 53, no. 2, pp. 963–983, 2015.

[103] M. Garavello, K. Han, and B. Piccoli, *Models for vehicular traffic on networks.* American Institute of Mathematical Sciences (AIMS), Springfield, MO, 2016, vol. 9.

[104] A. Keimer and L. Pflug, “Existence, uniqueness and regularity results on nonlocal balance laws,” *Journal of Differential Equations*, vol. 263, no. 7, pp. 4023–4069, 2017.

[105] A. Keimer, L. Pflug, and M. Spinola, “Existence, uniqueness and regularity of multidimensional nonlocal balance laws with damping,” *Journal of Mathematical Analysis and Applications*, vol. 466, no. 1, pp. 18–55, 2018.

[106] A. Bressan, *Hyperbolic systems of conservation laws: the one-dimensional Cauchy problem.* Oxford University Press on Demand, 2000, vol. 20.

[107] S. Blandin and P. Goatin, “Well-posedness of a conservation law with non-local flux arising in traffic flow modeling,” *Numerische Mathematik*, vol. 132, no. 2, pp. 217–241, 2016.

[108] P. Goatin and S. Scialanga, "Well-posedness and finite volume approximations of the lwr traffic flow model with non-local velocity," 2016.

[109] P. E. Kloeden and T. Lorenz, "Nonlocal multi-scale traffic flow models: Analysis beyond vector spaces," *Bulletin of Mathematical Sciences*, vol. 6, no. 3, pp. 453–514, 2016.

[110] F. A. Chiarello and P. Goatin, "Non-local multi-class traffic flow models," 2018.

[111] Y. Lee, "Thresholds for shock formation in traffic flow models with nonlocal-concave-convex flux," *Journal of Differential Equations*, vol. 266, no. 1, pp. 580–599, 2019.

[112] J. Ridder and W. Shen, "Traveling waves for nonlocal models of traffic flow," *arXiv preprint arXiv:1808.03734*, 2018.

[113] W. Shen, "Traveling waves for conservation laws with nonlocal flux for traffic flow on rough roads," *arXiv preprint arXiv:1809.02998*, 2018.

[114] D. Armbruster, D. E. Marthaler, C. Ringhofer, K. Kempf, and T.-C. Jo, "A continuum model for a re-entrant factory," *Operations research*, vol. 54, no. 5, pp. 933–950, 2006.

[115] J.-M. Coron, Z. Wang, *et al.*, "Analysis of a conservation law modeling a highly re-entrant manufacturing system," *arXiv preprint arXiv:0907.1274*, 2009.

[116] P. Shang and Z. Wang, "Analysis and control of a scalar conservation law modeling a highly re-entrant manufacturing system," *Journal of Differential Equations*, vol. 250, no. 2, pp. 949–982, 2011.

[117] M. Gugat, A. Keimer, G. Leugering, and Z. Wang, "Analysis of a system of nonlocal conservation laws for multi-commodity flow on networks," 2016.

[118] A. Keimer, G. Leugering, and T. Sarkar, "Analysis of a system of nonlocal balance laws with weighted work in progress," *Journal of Hyperbolic Differential Equations*, vol. 15, no. 03, pp. 375–406, 2018.

[119] E. Rossi, J. Weißen, P. Goatin, and S. Göttlich, "Well-posedness of a non-local model for material flow on conveyor belts," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 54, no. 2, pp. 679–704, 2020.

[120] F. A. Chiarello, P. Goatin, and L. M. Villada, "Lagrangian antidiffusive remap schemes for non-local multi-class traffic flow models," *Computational and Applied Mathematics*, vol. 39, no. 2, pp. 1–22, 2020.

[121] R. M. Colombo, F. Marcellini, and E. Rossi, "Biological and industrial models motivating nonlocal conservation laws: A review of analytic and numerical results," *Networks & Heterogeneous Media*, vol. 11, no. 1, p. 49, 2016.

[122] R. M. Colombo and M. Lécureux-Mercier, "Nonlocal crowd dynamics models for several populations," *Acta Mathematica Scientia*, vol. 32, no. 1, pp. 177–196, 2012.

[123] A. Keimer, A. M. Bayen, A. Hayat, B. Piccoli, S. Albeaik, S. McQuade, X. Gong, and Y. You, "A coupled pde-ode mean-field equation for traffic flow," 2020.

[124] G. Crippa and M. Lécureux-Mercier, "Existence and uniqueness of measure solutions for a system of continuity equations with non-local flow," *Nonlinear Differential Equations and Applications NoDEA*, vol. 20, no. 3, pp. 523–537, 2013.

[125] X. Gong and M. Kawski, "Weak measure-valued solutions of a nonlinear hyperbolic conservation law," *arXiv preprint arXiv:1903.00797*, 2019.

[126] B. Delaunay *et al.*, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.

[127] L. P. Chew, "Constrained delaunay triangulations," *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1989.

[128] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[129] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming meshes that split and merge," in *ACM SIGGRAPH 2009 papers*, 2009, pp. 1–10.

[130] A. Zaharescu, E. Boyer, and R. Horaud, "Transformesh: A topology-adaptive mesh-based approach to surface evolution," in *Asian Conference on Computer Vision*, Springer, 2007, pp. 166–175.

[131] S.-W. Cheng and J. Jin, "Deforming surface meshes," in *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, Springer, 2015, pp. 69–89.

[132] S.-W. Cheng and T. K. Dey, "Maintaining deforming surface meshes," in *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2008, pp. 112–121.

[133] S.-W. Cheng and J. Jin, "Edge flips and deforming surface meshes," in *Proceedings of the twenty-seventh annual symposium on Computational geometry*, 2011, pp. 331–340.

[134] V. Garikapati, "Modeling and simulation of automated mobility districts," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2018.

[135] ——, "Optimizing fleet operations in automated mobility districts: Serving on-demand mobility with automated electric shuttles," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2019.

[136] Y. Chen, S. Young, X. Qi, and J. Gonder, "Estimate of fuel consumption and ghg emission impact from an automated mobility district," in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*, IEEE, 2015, pp. 271–278.

[137] Y. Hou, S. E. Young, V. Garikapati, Y. Chen, and L. Zhu, "Initial assessment and modeling framework development for automated mobility districts," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2018.

[138] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[139] I. H. Zohdy, R. K. Kamalanathsharma, and H. Rakha, "Intersection management for autonomous vehicles using icacc," in *2012 15th international IEEE conference on intelligent transportation systems*, IEEE, 2012, pp. 1109–1114.

[140] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Optimal coordination of automated vehicles at intersections: Theory and experiments," *IEEE Transactions on Control Systems Technology*, 2018.

[141] C. Yu, Y. Feng, H. X. Liu, W. Ma, and X. Yang, "Corridor level cooperative trajectory optimization with connected and automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 105, pp. 405–421, 2019.

[142] S.-H. Lin and T.-Y. Ho, "Autonomous vehicle routing in multiple intersections," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ACM, 2019, pp. 585–590.

[143] M. Hausknecht, T.-C. Au, and P. Stone, "Autonomous intersection management: Multi-intersection optimization," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 4581–4586.

[144] M. Pavone, "Autonomous mobility-on-demand systems for future urban mobility," in *Autonomes Fahren*, Springer, 2015, pp. 399–416.

[145] K. Spieser, K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone, "Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in singapore," in *Road vehicle automation*, Springer, 2014, pp. 229–245.

[146] Z. Chen, F. He, Y. Yin, and Y. Du, "Optimal design of autonomous vehicle zones in transportation networks," *Transportation Research Part B: Methodological*, vol. 99, pp. 44–61, 2017.

[147] L. Conceição, G. Correia, and J. P. Tavares, "The deployment of automated vehicles in urban transport systems: A methodology to design dedicated zones," *Transportation Research Procedia*, vol. 27, pp. 230–237, 2017.

[148] A. M. Bayen, "Computational control of networks of dynamical systems: Application to the national airspace system," PhD thesis, stanford university, 2003.

[149] P. K. Menon, G. D. Sweriduk, and K. D. Bilimoria, "New approach for modeling, analysis, and control of air traffic flow," *Journal of guidance, control, and dynamics*, vol. 27, no. 5, pp. 737–744, 2004.

[150] D. Chen, M. Hu, H. Zhang, J. Yin, and K. Han, "A network based dynamic air traffic flow model for en route airspace system traffic flow optimization," *Transportation Research Part E: Logistics and Transportation Review*, vol. 106, pp. 1–19, 2017.

[151] D. B. Poudel, "Coordinating hundreds of cooperative, autonomous robots in a warehouse," *Jan*, vol. 27, pp. 1–13, 2013.

[152] V. Digani, "Traffic coordination for agv systems: An ensemble modeling approach," PhD thesis, Mar. 2016. DOI: 10.13140/RG.2.2.28743.50089.